Handbuch

TC3 OPC-UA

TwinCAT

Version: 1.6
Datum: 25.08.2015
Bestell-Nr.: TF6100

BECKHOFF

Inhaltsverzeichnis

1 Vorwort 5
1.1 Hinweise zur Dokumentation 5
1.2 Sicherheitshinweise 6

2 Übersicht 7
2.1 Szenarien 7
2.2 Versionsinformationen 10
2.3 Anwendungsbeispiele 11
2.3.1 Nachbearbeitung in der Cloud 11

3 Installation 17
3.1 Systemvoraussetzungen 17
3.2 Installation 18
3.3 Installation Windows CE 21
3.4 Installation Tc2 23
3.5 Installation Windows CE Tc2 26
3.6 Lizensierung 31

4 Konfiguration 36
4.1 Server 36
4.1.1 Übersicht 36
4.1.2 Quick Start 37
4.1.3 SPS 39
4.1.4 C++ 61
4.1.5 I/O 68
4.1.6 Matlab/Simulink 71
4.1.7 Dateiübertragung 74
4.1.8 Sicherheit 75
4.1.9 Verschiedenes 78
4.2 Client 86
4.2.1 Übersicht 86
4.2.2 Best practice 87
4.2.3 Sicherheit 101
4.3 Konfigurator 103
4.3.1 Übersicht 103
4.3.2 Data Access 104
4.3.3 Historical Access 105
4.3.4 Sicherheit 106
4.3.5 Online Panel 107
4.4 Sample Client 109
4.4.1 Übersicht 109
4.4.2 Sichere Verbindung mit OPC-UA Server herstellen 110
4.4.3 Im UA-Namensraum browsen 113
4.4.4 Watchliste verwenden 114

5 SPS-Bibliotheken 116
5.1 Tcx_PLCopen_OpcUa 116
5.1.1 Funktionsbausteine 116
5.1.2 Datentypen 126

6 Beispiele 132
6.1 Übersicht 132

7 Anhang 133
7.1 OPC-UA Server Fehlercodes 133

7.2 OPC-UA Client Fehlercodes 133
7.3 ADS Return Codes 135
8 Third Party Beispiele 139
8.1 Beispiel OPC UA Client von Inosoft (Third Party) 139

1 Vorwort

1.1 Hinweise zur Dokumentation

Diese Beschreibung wendet sich ausschließlich an ausgebildetes Fachpersonal der Steuerungs- und Automatisierungstechnik, das mit den geltenden nationalen Normen vertraut ist.
Zur Installation und Inbetriebnahme der Komponenten ist die Beachtung der nachfolgenden Hinweise und Erklärungen unbedingt notwendig.

Das Fachpersonal hat sicherzustellen, dass die Anwendung bzw. der Einsatz der beschriebenen Produkte alle Sicherheitsanforderungen, einschließlich sämtlicher anwendbaren Gesetze, Vorschriften, Bestimmungen und Normen erfüllt.

Disclaimer

Diese Dokumentation wurde sorgfältig erstellt. Die beschriebenen Produkte werden jedoch ständig weiter entwickelt.
Deshalb ist die Dokumentation nicht in jedem Fall vollständig auf die Übereinstimmung mit den beschriebenen Leistungsdaten, Normen oder sonstigen Merkmalen geprüft.
Falls sie technische oder redaktionelle Fehler enthält, behalten wir uns das Recht vor, Änderungen jederzeit und ohne Ankündigung vorzunehmen.
Aus den Angaben, Abbildungen und Beschreibungen in dieser Dokumentation können keine Ansprüche auf Änderung bereits gelieferter Produkte geltend gemacht werden.

Marken

Beckhoff®, TwinCAT®, EtherCAT®, Safety over EtherCAT®, TwinSAFE®, XFC®und XTS® sind eingetragene und lizenzierte Marken der Beckhoff Automation GmbH.
Die Verwendung anderer in dieser Dokumentation enthaltenen Marken oder Kennzeichen durch Dritte kann zu einer Verletzung von Rechten der Inhaber der entsprechenden Bezeichnungen führen.

Patente

Die EtherCAT Technologie ist patentrechtlich geschützt, insbesondere durch folgende Anmeldungen und Patente:
EP1590927, EP1789857, DE102004044764, DE102007017835
mit den entsprechenden Anmeldungen und Eintragungen in verschiedenen anderen Ländern.

Die TwinCAT Technologie ist patentrechtlich geschützt, insbesondere durch folgende Anmeldungen und Patente:
EP0851348, US6167425 mit den entsprechenden Anmeldungen und Eintragungen in verschiedenen anderen Ländern.

EtherCAT® ist eine eingetragene Marke und patentierte Technologie lizensiert durch die Beckhoff Automation GmbH, Deutschland

Copyright

© Beckhoff Automation GmbH & Co. KG, Deutschland.
Weitergabe sowie Vervielfältigung dieses Dokuments, Verwertung und Mitteilung seines Inhalts sind verboten, soweit nicht ausdrücklich gestattet.
Zuwiderhandlungen verpflichten zu Schadenersatz. Alle Rechte für den Fall der Patent-, Gebrauchsmuster- oder Geschmacksmustereintragung vorbehalten.

1.2 Sicherheitshinweise

Sicherheitsbestimmungen

Beachten Sie die folgenden Sicherheitshinweise und Erklärungen!
Produktspezifische Sicherheitshinweise finden Sie auf den folgenden Seiten oder in den Bereichen Montage, Verdrahtung, Inbetriebnahme usw.

Haftungsausschluss

Die gesamten Komponenten werden je nach Anwendungsbestimmungen in bestimmten Hard- und Software-Konfigurationen ausgeliefert. Änderungen der Hard- oder Software-Konfiguration, die über die dokumentierten Möglichkeiten hinausgehen, sind unzulässig und bewirken den Haftungsausschluss der Beckhoff Automation GmbH & Co. KG.

Qualifikation des Personals

Diese Beschreibung wendet sich ausschließlich an ausgebildetes Fachpersonal der Steuerungs-, Automatisierungs- und Antriebstechnik, das mit den geltenden Normen vertraut ist.

Erklärung der Symbole

In der vorliegenden Dokumentation werden die folgenden Symbole mit einem nebenstehenden Sicherheitshinweis oder Hinweistext verwendet. Die Sicherheitshinweise sind aufmerksam zu lesen und unbedingt zu befolgen!

GEFAHR	**Akute Verletzungsgefahr!** Wenn der Sicherheitshinweis neben diesem Symbol nicht beachtet wird, besteht unmittelbare Gefahr für Leben und Gesundheit von Personen!
WARNUNG	**Verletzungsgefahr!** Wenn der Sicherheitshinweis neben diesem Symbol nicht beachtet wird, besteht Gefahr für Leben und Gesundheit von Personen!
VORSICHT	**Schädigung von Personen!** Wenn der Sicherheitshinweis neben diesem Symbol nicht beachtet wird, können Personen geschädigt werden!
Achtung	**Schädigung von Umwelt oder Geräten** Wenn der Hinweis neben diesem Symbol nicht beachtet wird, können Umwelt oder Geräte geschädigt werden.
Hinweis	**Tipp oder Fingerzeig** Dieses Symbol kennzeichnet Informationen, die zum besseren Verständnis beitragen.

2 Übersicht

OPC Unified Architecture (OPC-UA) ist die nächste Generation des klassischen OPC-Standards. Es handelt sich hierbei um ein weltweit standardisiertes Kommunikationsprotokoll, mit dessen Hilfe sich Maschinendaten sowohl Hersteller-, als auch Plattformunabhängig austauschen lassen. Hierbei integriert OPC-UA bereits direkt im Protokoll gängige Sicherheitsstandards. Ein weiterer großer Vorteil von UA gegenüber dem klassischen OPC-Standard ist hierbei auch die Unabhängigkeit vom COM/DCOM-System. Für detaillierte Informationen zu OPC-UA sei hiermit auf die Webseiten der OPC Foundation verwiesen.

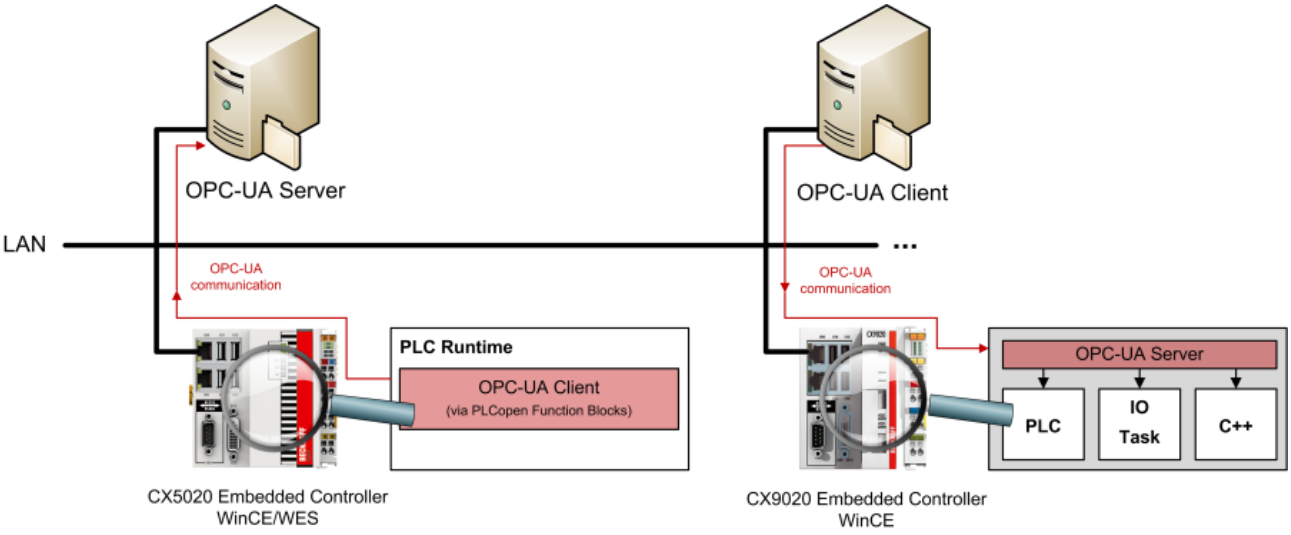
Komponenten

Das Produkt besteht seit der Version 1.6.80 aus den folgenden Komponenten, sowohl unter Win32/x64 als auch Windows CE-basierten Systemen:

- OPC-UA Server [▶ 36]: bietet eine OPC-UA Server Schnittstelle um UA-Clients den Zugriff auf die TwinCAT-Welt zu ermöglichen
- OPC-UA Client: stellt eine OPC-UA Client Funktionalität zur Verfügung, um – basierend auf der Spezifikation “PLCopen OPC-UA Client Functions blocks” - mit anderen OPC-UA Servern kommunizieren zu können. Die entsprechenden Funktionsbausteine werden im Rahmen der Tcx_PLCopen_OpcUa Bibliothek automatisch mit durch das Setup installiert und im TwinCAT PLC Library Repository registriert.

Unter Windows XP-basierten Systemen werden des Weiteren noch die folgenden UA-Tools mit ausgeliefert:

- OPC-UA Configurator [▶ 103]: eine grafische, .NET-basierende Oberfläche zur Konfiguration des OPC-UA Servers
- OPC-UA SampleClient [▶ 109]: eine OPC-UA Client Beispiel-Implementierung in Form einer grafischen .NET-Anwendung zum Navigieren durch den OPC-UA Namensraums. Kann zum Testen der OPC-UA Server Installation verwendet werden.



2.1 Szenarien

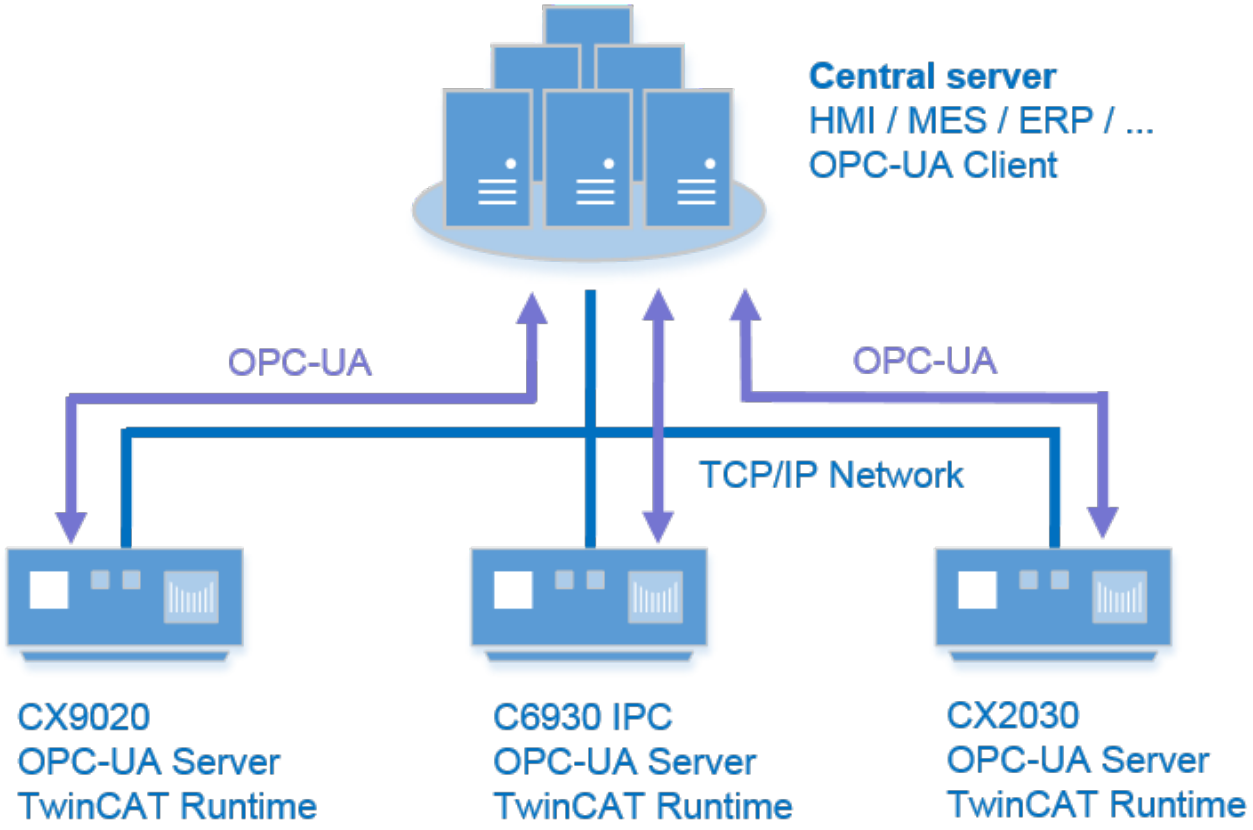
Die folgenden Artikel der Dokumentation erläutern einige Szenarien, gemäß derer die Komponenten je nach Infrastruktur des Kunden und Anwendungsfall implementiert und verwendet werden können. Er besteht aus folgenden Themen:

- OPC-UA Server: In Industrie-PC oder Embedded-PC integriert
- OPC-UA Server: Läuft auf einem Zentralrechner mit Verbindung zu ferner(n) TwinCAT Laufzeit(en).
- OPC-UA Server: Zugriff auf BC Controller

OPC-UA Server: In Industrie-PC oder Embedded-PC integriert

Einer der größten Vorteile des TwinCAT OPC-UA Servers besteht darin, dass er selbst in die kleinste Embedded-Plattform, z.B. die CX8000 Baureihe, integriert werden kann. Dank dieser Integration ist das allgemeine Handling sehr einfach und komfortabel. OPC-UA Clients, z.B. HMI oder MES/ERP Systeme, können eine Verbindung mit dem OPC-UA Server herstellen und aus der TwinCAT Laufzeit stammende Symbolinformationen lesen oder schreiben.

Hinweis	**Empfohlen** Dies ist das empfohlene Szenario, gemäß dem der TwinCAT OPC-UA Server unter normalen Umständen eingesetzt werden sollte.



Auf dem zentralen Server laufen die folgenden Softwarekomponenten und Konfigurationen.

- Dritter OPC-UA Client, der z.B. ein HMI-, MES-, oder ERP-System sein kann.

Auf dem Industrie-PC oder Embedded-PC laufen die folgenden Softwarekomponenten und Konfigurationen.

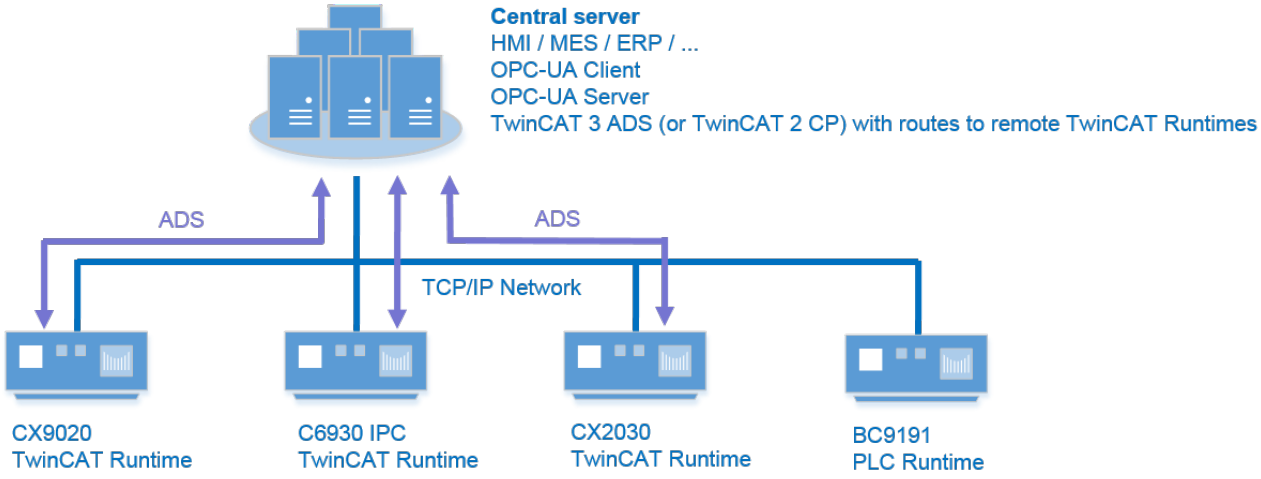
- Der TwinCAT OPC-UA Server stellt automatisch eine Verbindung mit der ersten lokalen TwinCAT SPS Laufzeit her.
- TwinCAT Laufzeit

Dieses Szenario hat die folgenden Vorteile:

- Die Netzwerknutzung ist optimiert, weil sie auf OPC-UA Kommunikationstechniken, z.B. OPC-UA Registrierungen aufbaut.
- Die Speichernutzung ist dezentral – jedes Gerät ist ausschließlich für den eigenen Speicherbedarf zuständig.
- Sicherheit – OPC-UA verfügt über direkt ins Protokoll integrierte Sicherheitsmechanismen. Dies ist möglicherweise sehr praktisch, wenn einer der Industrie-PCs oder Embedded-PC übers Internet verbunden ist.

OPC-UA Server: Läuft auf einem Zentralrechner mit Verbindung zu ferner(n) TwinCAT Laufzeit(en).

Dieses Szenario beschreibt die klassische Implementierung von OPC-Server. Server, die die alte OPC-DA Technologie verwenden, waren häufig auf einem zentralen Server implementiert, statt auf dem Industrie-PC, auf dem die TwinCAT Laufzeit ausgeführt wird, um die DCOM Konfigurationen zu vermeiden. (Zur Erinnerung: Anders als OPC-UA basierte OPC-DA auf COM/DCOM Technologien)



Auf dem zentralen Server laufen die folgenden Softwarekomponenten und Konfigurationen.

- TwinCAT 3 ADS (oder TwinCAT 2 CP) für die erforderliche ADS Konnektivität
- TwinCAT OPC-UA Server, mit Datenzugriffsgeräten für ferne TwinCAT Laufzeiten konfiguriert
- ADS Routen zu fernen TwinCAT Laufzeiten
- Symboldateien von jeder fernen TwinCAT Laufzeit
- OPC-UA Client, der z.B. ein HMI-, MES- oder ERP-System sein kann.

Auf dem Industrie-PC oder Embedded-PC laufen die folgenden Softwarekomponenten und Konfigurationen.

- TwinCAT Laufzeit
- ADS-Route zum zentralen Server

Auch wenn die Netzwerknutzung bei diesem Szenario noch gut sein mag, wird sie möglicherweise ansteigen, je mehr ferne TwinCAT Laufzeiten mit dem zentralen TwinCAT OPC-UA Server verbunden werden. Der OPC-UA Server verwendet fortschrittliche ADS Aufzeichnungstechniken, um die Symbolwerte von der TwinCAT Laufzeit abzufragen. Je mehr Symbole vorhanden sind, je mehr zyklische Abfragen werden unterwegs sein. Demzufolge findet die optimierte OPC-UA Kommunikation nur lokal auf dem zentralen Server statt (zwischen OPC-UA Client und OPC-UA Server)!

Demzufolge weist dieses Szenario im Vergleich zum ersten die folgenden Nachteile auf:

- Die Netzwerknutzung kann sehr hoch sein, in Abhängigkeit der Anzahl vorhandener Geräte und Symbole.
- Der Speicherbedarf auf dem zentralen Server ist sehr hoch, weil der OPC-UA Server Symbolinformationen von JEDER TwinCAT Laufzeit importieren muss.
- Keine Sicherheit zwischen zentralem Server und der TwinCAT Laufzeit dank Datenverschlüsselung – ADS wurde auf hohe Leistung ausgelegt und bietet keine Datenverschlüsselung.
- Die Notwendigkeit, bei jeder SPS-Programmänderung Symboldateien zwischen TwinCAT Laufzeit und zentralem Server auszutauschen.

OPC-UA Server: Zugriff auf BC Controller

Auch wenn unsere kleinen BC Controller nicht in der Lage sind, einen eigenen OPC-UA Server auszuführen, können Sie trotzdem Zugriff auf die auf einem fernen BC Controller, z.B. ein BC9191, ausgeführten SPS-Laufzeit haben und deren Symbolwerte über OPC-UA veröffentlichen. Bei diesem Szenario müssen Sie den OPC-UA Server zusammen mit einer TwinCAT 3 ADS (oder TwinCAT 2 CP) auf einem anderen Rechner ausführen und das Szenario auf die gleiche Weise wie Szenario 2 konfigurieren.

2.2 Versionsinformationen

Die folgenden Tabellen zeigen, welche Versionen eines Einzelprodukts in dem jeweiligen TS6100/TF6100 Setup enthalten sind.

Hinweis	**Wichtige Hinweise** Wichtige Hinweise können Sie der Datei „readme.txt“ entnehmen, die Sie zusammen mit dem Setup erhalten, insbesondere, wenn Sie von älteren Versionen einer enthaltenen Softwarekomponente upgraden (z. B. beim Upgrade der TcOpcUaServer.exe von der Version 2.0.x auf 2.1.x).

Tab. 1: Setup Version 3.2.1

Dateiname	Version
TwinCAT OPC-UA Server	2.1.0.28
TwinCAT OPC-UA Client	2.1.0.18
TwinCAT OPC-UA Konfigurator	2.1.0.29
TwinCAT OPC-UA Beispiel-Client	1.5.0.0
SPS-Bibliothek Tc2_PLCopen_OpcUa (nur TS6100)	1.0.1.0
SPS-Bibliothek Tc3_PLCopen_OpcUa (nur TF6100)	3.1.2.0

Tab. 2: Setup Version 3.1.20

Dateiname	Version
TwinCAT OPC-UA Server	2.0.0.72
TwinCAT OPC-UA Client	2.1.0.18
TwinCAT OPC-UA Konfigurator	2.1.0.24
TwinCAT OPC-UA Beispiel-Client	1.5.0.0
SPS-Bibliothek Tc2_PLCopen_OpcUa (nur TS6100)	1.0.1.0
SPS-Bibliothek Tc3_PLCopen_OpcUa (nur TF6100)	3.1.2.0

Tab. 3: Setup Version 3.1.18

Dateiname	Version
TwinCAT OPC-UA Server	2.0.0.71
TwinCAT OPC-UA Client	2.1.0.18
TwinCAT OPC-UA Konfigurator	2.1.0.23
TwinCAT OPC-UA Beispiel-Client	1.5.0.0
SPS-Bibliothek Tc2_PLCopen_OpcUa (nur TS6100)	1.0.1.0
SPS-Bibliothek Tc3_PLCopen_OpcUa (nur TF6100)	3.1.2.0

Tab. 4: Setup Version 3.1.16

Dateiname	Version
TwinCAT OPC-UA Server	2.0.0.70
TwinCAT OPC-UA Client	2.1.0.18
TwinCAT OPC-UA Konfigurator	2.1.0.22
TwinCAT OPC-UA Beispiel-Client	1.5.0.0
SPS-Bibliothek Tc2_PLCopen_OpcUa (nur TS6100)	1.0.1.0
SPS-Bibliothek Tc3_PLCopen_OpcUa (nur TF6100)	3.1.2.0

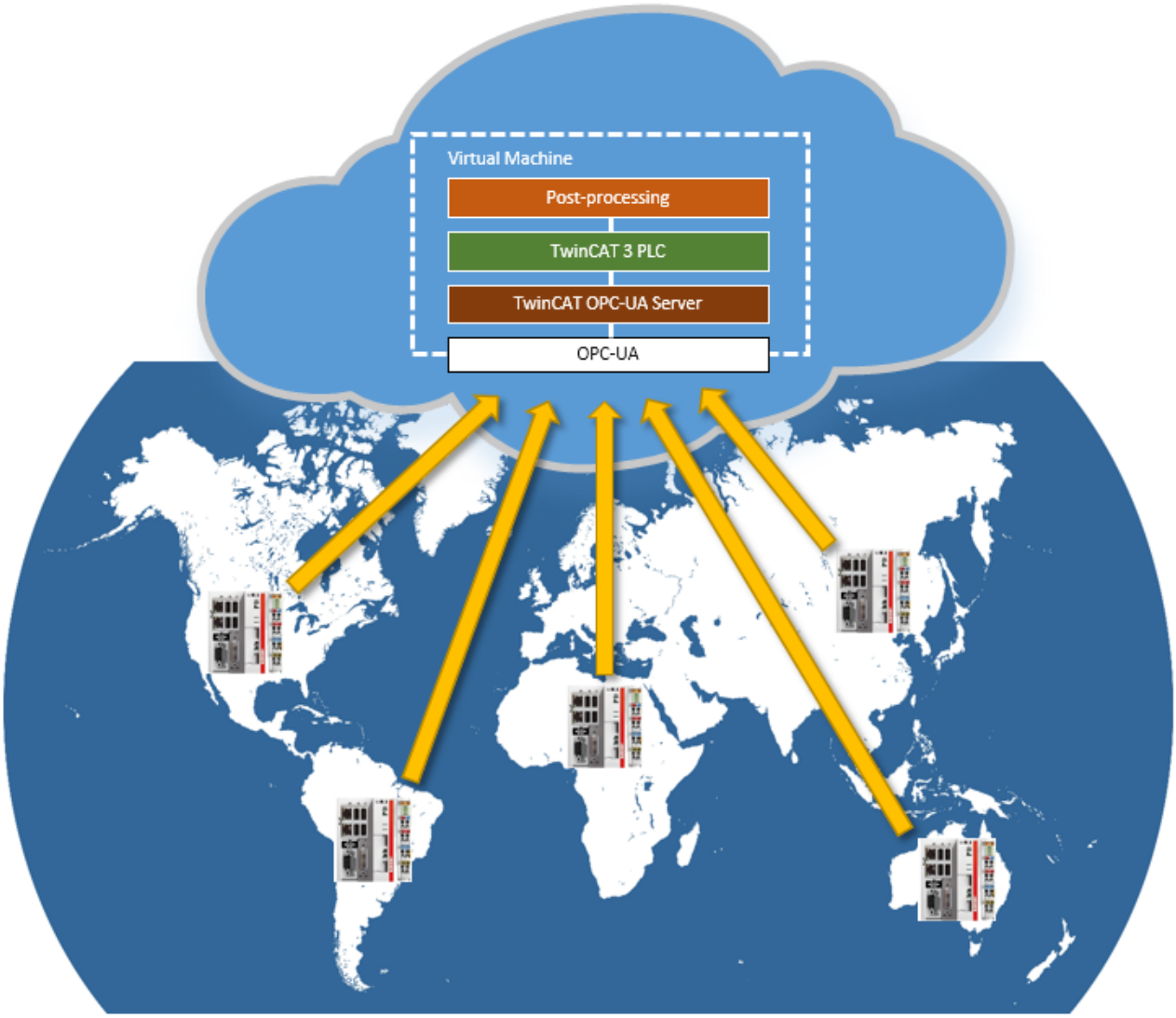
Tab. 5: Setup Version 3.1.14

Dateiname	Version
TwinCAT OPC-UA Server	2.0.0.67
TwinCAT OPC-UA Client	2.1.0.18
TwinCAT OPC-UA Konfigurator	2.1.0.22
TwinCAT OPC-UA Beispiel-Client	1.5.0.0
SPS-Bibliothek Tc2_PLCopen_OpcUa (nur TS6100)	1.0.1.0
SPS-Bibliothek Tc3_PLCopen_OpcUa (nur TF6100)	3.1.2.0

2.3 Anwendungsbeispiele

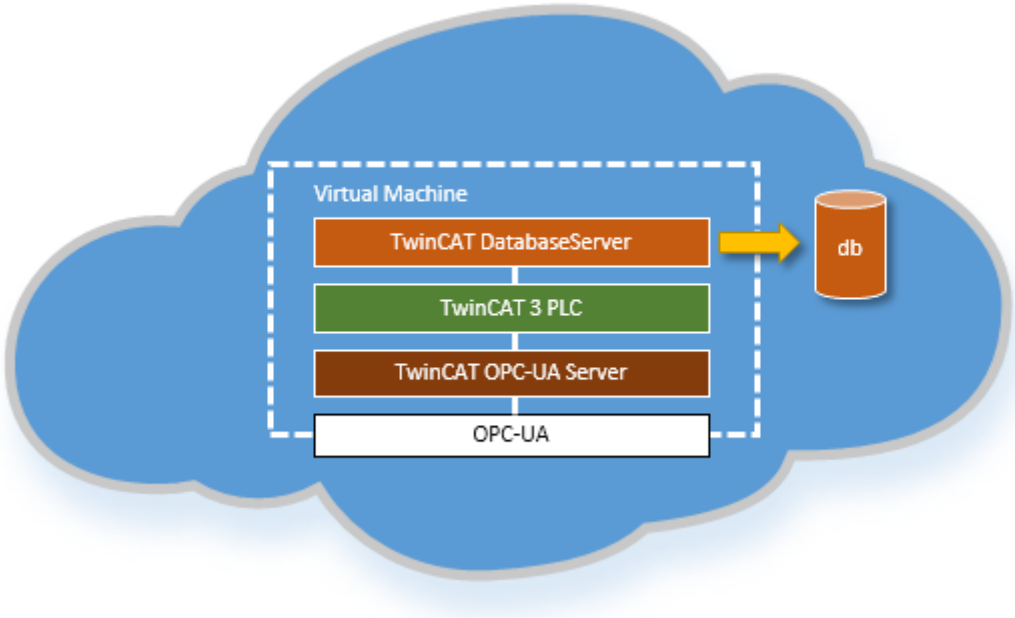
2.3.1 Nachbearbeitung in der Cloud

Das Gesamtkonzept zur Bereitstellung einer generischen, standardisierten und aufrufbaren Schnittstelle in der Cloud, die eingehende Daten zur weiteren Analyse akzeptiert, basiert – neben den dezentralisierten Clients, die diesen Service aufrufen – auf einem Endpunkt in der Cloud, der entweder eine Windows-basierte virtuelle Maschine oder ein echte Hardware ist, auf der ein Windows Betriebssystem läuft. In dieser Dokumentation wird dieses Gerät im Allgemeinen als „OPC-UA Server" (in der Cloud) bezeichnet.



Anwendungsbeispiel: Daten-Logger

Das Gesamtdokument zeigt, wie auf Rechenressourcen in der Cloud zugegriffen werden kann und dezentrale TwinCAT SPS-Geräte angeschlossen werden können, um diese Ressourcen zu nutzen. Als erstes Anwendungsbeispiel kann dieser Anwendungsfall auf ein typisches Datenspeicherungs-Szenario angewandt werden, in dem dezentrale TwinCAT SPS-Geräte Daten an die Cloud senden, in der die eingehenden Daten in einer SQL-Datenbank über den TwinCAT Database Server gespeichert werden. Obwohl das allgemeine Szenario ebenfalls das Hin- und Zurücksenden von Daten umfasst (d.h. Senden von Parametern an die Cloud, Ausführung der Berechnungen in der Cloud und dann Rücksendung der berechneten Ergebnisse), erfordert dieses spezifische Szenario nicht die Rückgabe von Ergebnissen aus der Cloud.



Systemanforderungen: OPC-UA Server („Die Cloud“)

Die Cloud muss in der Lage sein, eine der folgenden Betriebssystemvarianten zu hosten. Beachten Sie dabei, dass die Verwendung von Virtualisierungstechnologie eine kostenintensivere Lösung für 64-Bit Betriebssysteme erfordert.

Gerät ist eine virtuelle Maschine

- 32-Bit Windows Betriebssystem
- 64-Bit Windows Betriebssystem mit CPU Core Isolation (mindestens 2 Kerne erforderlich)

Gerät ist eine dedizierte Hardware

- 32-Bit Windows Betriebssystem
- 64-Bit Windows Betriebssystem

Zusätzlich müssen die folgenden Softwarekomponenten auf dem Gerät installiert werden:

- TwinCAT 3 XAR (nur Runtime) oder XAE (Runtime und Engineering)
- TwinCAT 3 Function TF6100 OPC-UA

Bitte beachten: Bei Installation eines kompletten TwinCAT 3 XAE könnten weitere Konfigurationseinstellungen sinnvoll sein, wie z. B. das Handling von Lizenzen oder die Möglichkeit zum Debuggen des Geräts direkt in der Cloud. Zum Speichern von Daten in einer zentralen Datenbank die von einem Client eingehen, müssen die folgenden Softwarekomponenten installiert sein:

- TwinCAT 3 Function TF6420 Database

Systemanforderungen: Dezentralisierter OPC-UA Client

Der dezentralisierte OPC-UA Client basiert auf einer TwinCAT 3 SPS-Laufzeit und kann daher auf jeder Hardwarekonfiguration gehostet werden, die den Betrieb einer TwinCAT 3 SPS unterstützt. Zudem muss die TwinCAT 3 Function TF6100 OPC-UA auf dem Client-Gerät installiert werden, um den TwinCAT OPC-UA Client nutzen zu können, um UA-Methoden auf dem Server in der Cloud auszuführen.

Technische Beschreibung

Die Realisierung eines OPC-UA Servers in der Cloud ist sehr anwendungsspezifisch. Beckhoff stellt eine allgemeine Architekturbeschreibung zur Verfügung, unabhängig von dem Anwendungsfall im Einzelnen. Als Anwendungsbeispiel beschreibt dieser Artikel häufig ein Datenaggregationsszenario, bei dem Prozessdaten von dezentralisierten Clients über OPC-UA zu einem zentralen Server übertragen werden, auf welchem die Daten in einer zentralen Datenbank angesammelt werden. Der Vorteil von OPC-UA besteht in diesem Fall nicht nur in der standardisierten Schnittstelle (und daher zur Nutzung dieses Konzepts für jede Art von OPC-UA Client), sondern auch in der Möglichkeit, den Kommunikationskanal zu sichern, indem der integrierte Sicherheitsmechanismus von OPC-UA eingesetzt wird.

Der TwinCAT OPC-UA Server wird verwendet, um die in der IEC61131-3 SPS-definierten Methoden dem OPC-UA Namensraum offenzulegen. Diese Methoden können von jedem TwinCAT OPC-UA Client (auf der Grundlage der PLCopen Funktionsblöcke) oder von 3rd-Party OPC-UA Clients aufgerufen werden.

Dezentralisierter OPC-UA Client

Der Hauptzweck des OPC-UA Clients besteht darin, Methoden auf dem Remote-Server aufzurufen. Der Client ist mit großer Wahrscheinlichkeit ein TwinCAT 3 Embedded Gerät, das den dezentralen Teil der Anwendung erfüllt. Dies würde für das Datenaggregations-Beispiel bedeuten, Prozessdaten abzufragen und sie durch Aufruf einer Methode an den Server zu senden.

Schnittstellenbeschreibung (zur Cloud)

Als Schnittstelle zu einem zentralisierten Cloud-System gibt es nur zwei vom Server bereitgestellte Methoden:

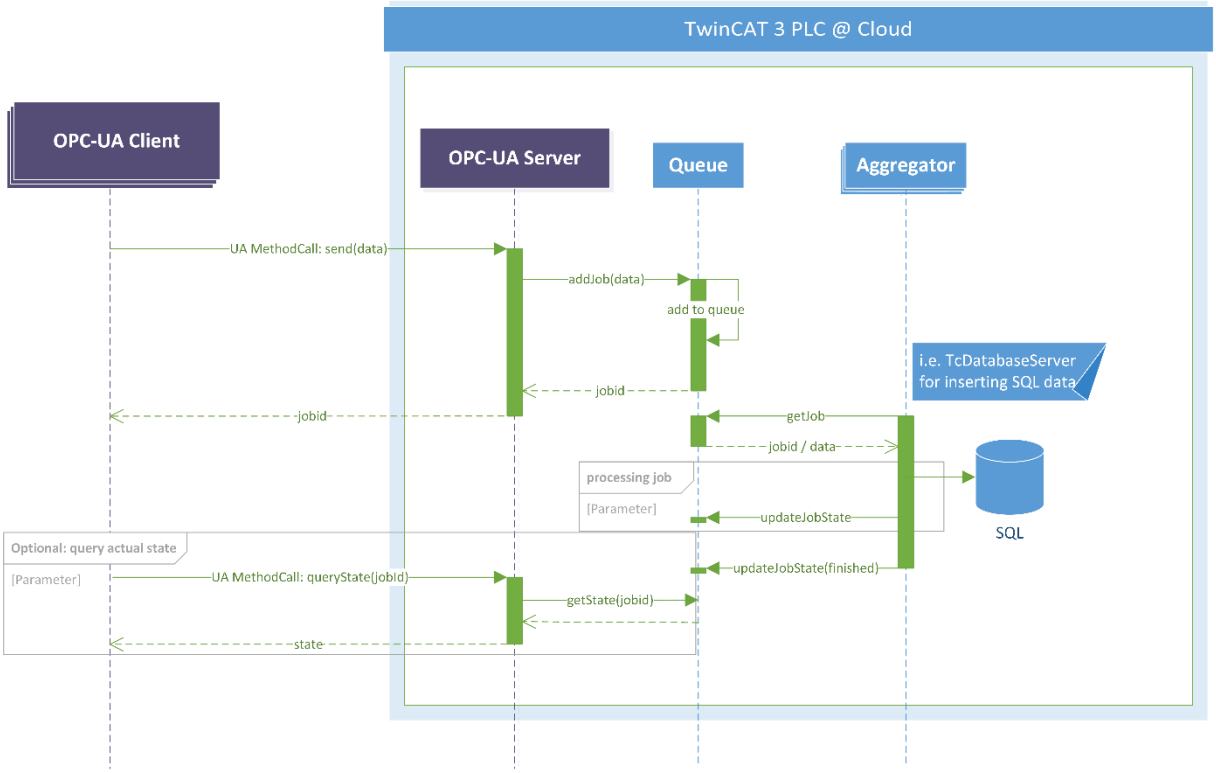
- int Send(data): Diese Methode überträgt die Daten über einen OPC-UA Methodenaufruf vor und gibt eine JobID zurück
- int QueryState(jobID): Unter Verwendung der zuvor abgerufenen JobID kann der Client zyklisch den aktuellen Status des Jobs abfragen. Dies könnte ausgelassen werden, wenn der Server in der Cloud alle Situationen bearbeiten kann oder der Client bei Problemen ohnehin nicht aushelfen kann.

Die Kommunikation zwischen Clients und der Cloud kann durch die eingebauten Sicherheitskonzepte von OPC-UA gesichert werden, z. B. durch die Verwendung von Client- und Server-Zertifikaten zur Verschlüsselung der Datenkommunikation. In der TF6100-Dokumentation erhalten Sie weitere Informationen.

Die Cloud

Der OPC-UA Server in der Cloud basiert auf TwinCAT 3 SPS-Methoden, die dem OPC-UA Namensraum offengelegt werden. Das SPS-Projekt enthält die folgenden drei Komponenten:

- MethodCall Provider: Diese Komponente bietet die zuvor erwähnte *Send()* Methode, bei der Daten gesammelt, eine JobID erstellt und die Daten als verschiedene Jobs in die Queue eingestellt werden. Darüber hinaus könnte es eine *QueryState()* Methode geben, um es OPC-UA Clients zu ermöglichen, den aktuellen Jobstatus abzufragen.
- Queue: Die Warteschlange speichert eine Liste der Jobs, die ausgeführt werden sollen. Jeder Job in der Warteschlange enthält die folgenden Informationen: {ID, Status, Data}. Neue Elemente (Jobs) werden durch den MethodCall Provider hinzugefügt und können durch diesen auch gelöscht werden. Die gesamte Warteschlange kann innerhalb des SPS-Programms auf einer Hash-Tabelle basieren.
- Aggregator: Die Anwendung, die die Jobs verarbeitet. Sie schaut zyklisch in der Queue, ob es zu bearbeitende Jobs gibt. Wenn dies der Fall ist, übernimmt sie den Job (stellt den Status auf „verarbeitet“) und beginnt mit der Verarbeitung des Jobs, z. B. der Verbindung zu einer Datenbank über die Funktionsblöcke des TwinCAT DatabaseServer. Beachten Sie, dass es mehr als einen Aggregator geben kann, um eine parallele Verarbeitung der Queue zu ermöglichen.



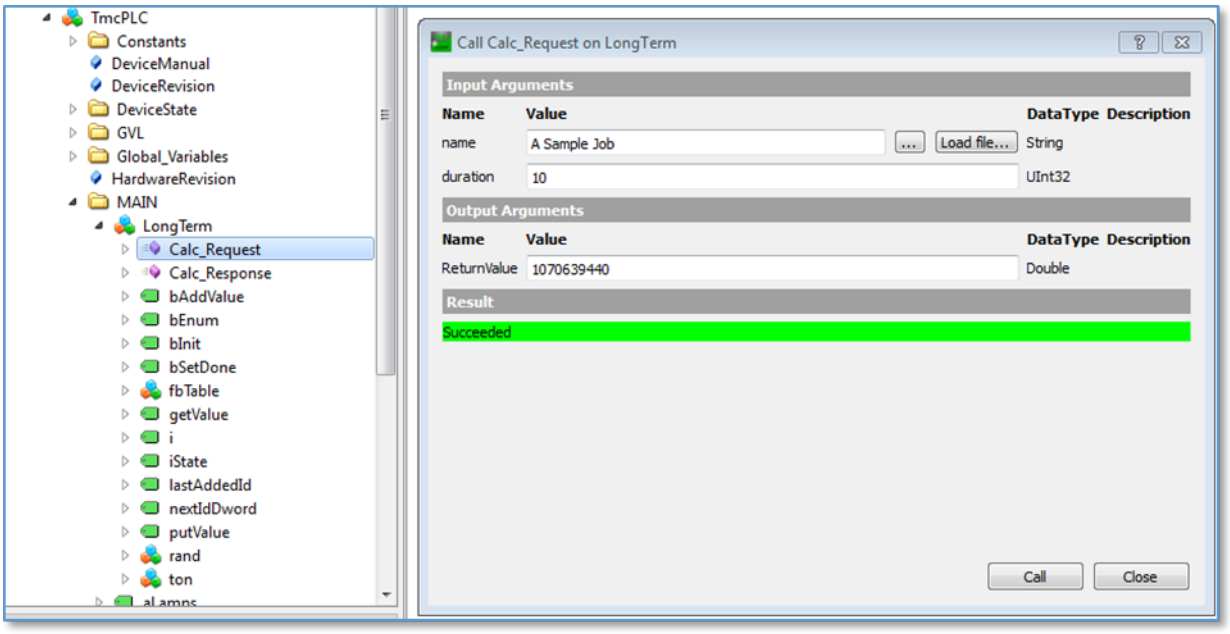
Beispiel

Das folgende Beispiel illustriert das Szenario durch die Ausführung einfacher Jobs: Jobs können dem Server von einem OPC-UA Client vorgelegt werden. In diesem Beispiel bestehen die Jobdaten aus einem definierbaren Zeitintervall, das vom Aggregator aufgenommen wird, um den Job fertigzustellen. Das Beispiel besteht aus den folgenden SPS-Komponenten auf dem OPC-UA Server:

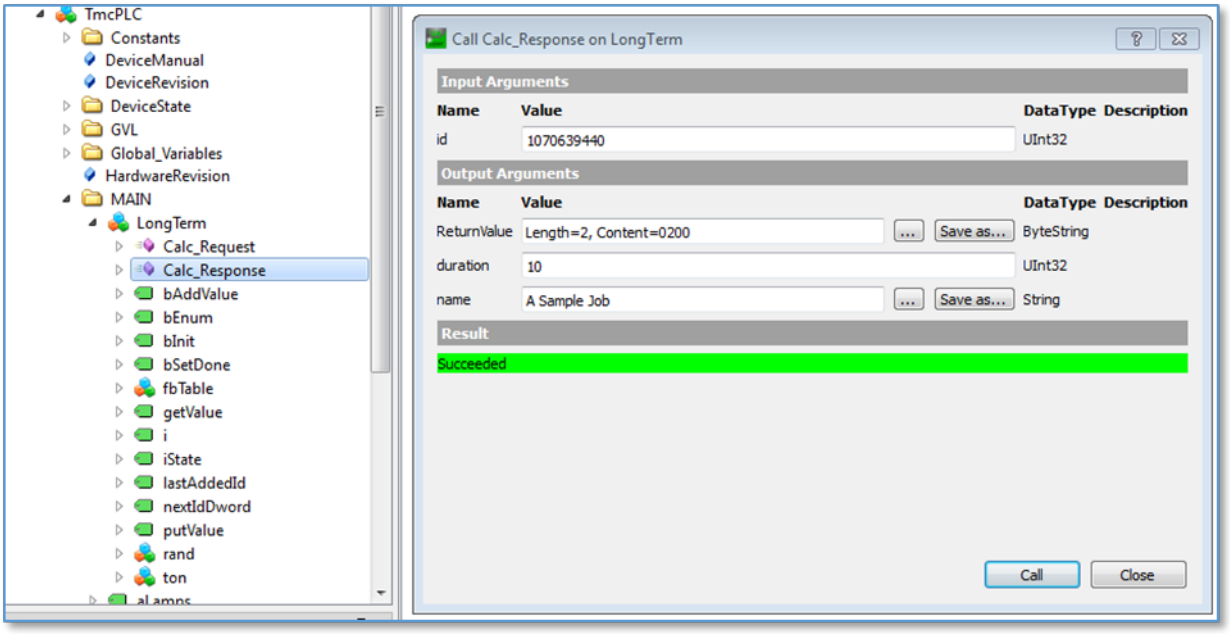
- ST_JobEntry: Repräsentiert einen Job in der Queue. Der Job besteht bei den Daten lediglich aus einem Namen und einer Zeitdauer. Eine Dauer definiert die Zeitspanne, die ein Job benötigt.
- E_JobState: Repräsentiert den Status eines Jobs. Die Beispielimplementierung definiert die folgenden Werte: QUEUED, PROCESSING, READY, FAILED und INVALID.
- LongTerm: Repräsentiert den MethodCall Server (besteht aus den Methoden Calc_Request und Calc_Response) sowie den Aggregator, der den Job verarbeitet (der im Funktionsblock implementiert ist)
- FB_SpecialHashTableCtrl: Repräsentiert die Queue in Form einer Hash-Tabelle, wie dies durch SPS-Beispiele vom Beckhoff Informationssystem wiedergegeben ist. Hier werden verschieden Methoden zum Umgang mit der Warteschlange gegeben (Hinzufügen, Zählen, GetFirst, GetNext, Lookup, Entfernen, Zurücksetzen).

Verwendung des Beispiels

Der UA Expert könnte als Beispiel-Client verwendet werden, um die Methoden aufzurufen, die vom OPC-UA Server in der Cloud bereitgestellt werden. Durch Aufruf der Methode Calc_Request erhält der Client eine JobID (oder 0 zur Anzeige eines Fehlers):



Indem die zurückgegebene JobId zur Methode Calc_Response gebracht wird, kann der Client den JobState abfragen. Schnell nach Calc_Request wird dies 0100 und später:



Zudem werden die vorher eingestellte Dauer und der Jobname als spätere Referenz zurückgegeben.

Installation

Das folgende Kapitel beschreibt die Installation und Konfiguration jeder Softwarekomponente, die auf dem Server in der Cloud erforderlich ist.

Nur Runtime

Wenn der Server nicht die Engineering Umgebung von TwinCAT 3 hostet, muss die TwinCAT 3 XAR Installation verwendet werden. Denken Sie daran, dass in diesem Fall das gesamte Handling zur ordnungsgemäßen Funktion mehrere Schritte involviert und die zukünftige Wartung ggf. schwieriger ist, da der XAR-Installation die Programmierungs- und Debugging-Umgebung fehlt. In diesem Szenario muss der Benutzer sicherstellen, dass die Pflege der ADS-Routen zwischen dem tatsächlichen Engineering-Computer (auf dem das SPS-Programm für den Server entwickelt wird) und dem Gerät, das den Server in der Cloud hostet, nicht nur den Download und das Debugging des SPS-Programms erlauben muss, sondern auch die

Aktivierung von Lizenzen für die TwinCAT 3 Laufzeit. Denken Sie daran, dass Sie die Firewall-Ports öffnen müssen, um ADS-Verkehr zu ermöglichen. In der ADS-Dokumentation erhalten Sie weitere Informationen. Zur einfacheren Handhabung der TwinCAT 3 Laufzeitlizenzen wurde das Tool TC3 License Request Generator entwickelt. Sie kann über den Beckhoff FTP-Server heruntergeladen werden und ermöglicht die Erstellung von License Request Files und den Import von License Response Files:

ftp.beckhoff.com/Software/TwinCAT/Unsupported_Utilities/TC3-LicenseGen/

Runtime und Engineering

Dies ist das empfohlene Setup-Szenario. Der Server in der Cloud hostet eine TwinCAT 3 Runtime und die entsprechenden Engineering Komponenten, um das Debuggen und eine einfachere Handhabung des Laufzeitsystems zu ermöglichen. In diesem Fall ist es erforderlich, TwinCAT 3 XAE auf dem System zu installieren.

Zusätzliche Software

Nach erfolgreicher Installation von TwinCAT XAE oder XAR müssen die folgenden Softwarekomponenten auf dem Gerät installiert und konfiguriert werden:

TF6100 OPC-UA

Die TF6100-Function installiert einen OPC-UA Server (und Client) auf dem Gerät. Der Server ist erforderlich, um den OPC-UA Clients die SPS-Methoden zur Verfügung zu stellen. Das TF6100-Setup ist gemäß der TF6100-Dokumentation zu installieren. Wenn das SPS-Programm in die Laufzeit heruntergeladen wird, importiert der OPC-UA Server automatisch die erste SPS-Laufzeit in seinen Namensraum. Sämtliche Variablen und Methoden der SPS, die als verfügbar über OPC-UA gekennzeichnet sind, werden in den OPC-UA Namensraum importiert und werden für Clients verfügbar. In der TF6100-Dokumentation erhalten Sie weitere Informationen.

TF6420 Database

Wenn der Server die von den Clients erhaltenen Daten in einer Datenbank speichern soll, muss die TF6420-Function verwendet werden. Sie bietet generische Funktionsblöcke für die TwinCAT 3 SPS für den Zugriff auf die Datenbank, z. B. um Werte in eine Datenbanktabelle einzufügen. Installieren Sie das TF6420-Setup und konsultieren Sie die TF6420-Dokumentation für weitere Informationen zur Nutzung der entsprechenden Funktionsblöcke.

Lizenzierung

Alle TwinCAT 3 Softwareprodukte erfordern eine Lizenz, um auf dem Server verfügbar zu sein. Die Lizenz kann entweder aus einer 7-Tage-Probelizenz oder aus einer uneingeschränkten Lizenz bestehen. Lizenzen können unter Verwendung des TwinCAT 3 XAE Lizenzmanagers aktiviert werden. Wenn eine TwinCAT 3 XAR Installation auf dem Server verwendet wird, verwenden Sie den TC3 License Generator, wie vorstehend beschrieben.

3 Installation

3.1 Systemvoraussetzungen

OPC-UA Server

- Betriebssysteme:
  - Windows XP Pro SP3
  - Windows 7 Pro (32-bit und 64-bit)
  - Windows XP Embedded
  - Windows Embedded Standard 2009
  - Windows Embedded 7
  - Windows Server 2008 R2
  - Windows Server 2012
  - Windows CE6
  - Windows CE7
- TwinCAT:
  - TwinCAT 3 ADS Build 4012 (oder höher)
  - TwinCAT 3 XAR Build 3100 (oder höher)
  - TwinCAT 3 XAE Build 3100 (oder höher)
  - TwinCAT 2

Eventuell auftretende Unterschiede zwischen diesen Betriebssystemvarianten werden in der Installationsanleitung beschrieben.

OPC-UA Client

- Betriebssysteme:
  - Windows XP Pro SP3
  - Windows 7 Pro (32-bit und 64-bit)
  - Windows XP Embedded
  - Windows Embedded Standard 2009
  - Windows Embedded 7
  - Windows Server 2008 R2
  - Windows Server 2012
  - Windows CE6
  - Windows CE7
- TwinCAT:
  - TwinCAT 3 ADS Build 4012 (oder höher)
  - TwinCAT 3 XAR Build 3100 (oder höher)
  - TwinCAT 3 XAE Build 3100 (oder höher)
  - TwinCAT 2.11 R3 Build 2245 (oder höher)

OPC-UA Konfigurator

- Betriebssysteme:
  - Windows XP Pro SP3
  - Windows 7 Pro (32-bit und 64-bit)
  - Windows XP Embedded

- Windows Embedded Standard 2009
- Windows Embedded 7
- Windows Server 2008 R2
- Windows Server 2012

- Sonstiges:
  - .NET Framework 2.0 SP1

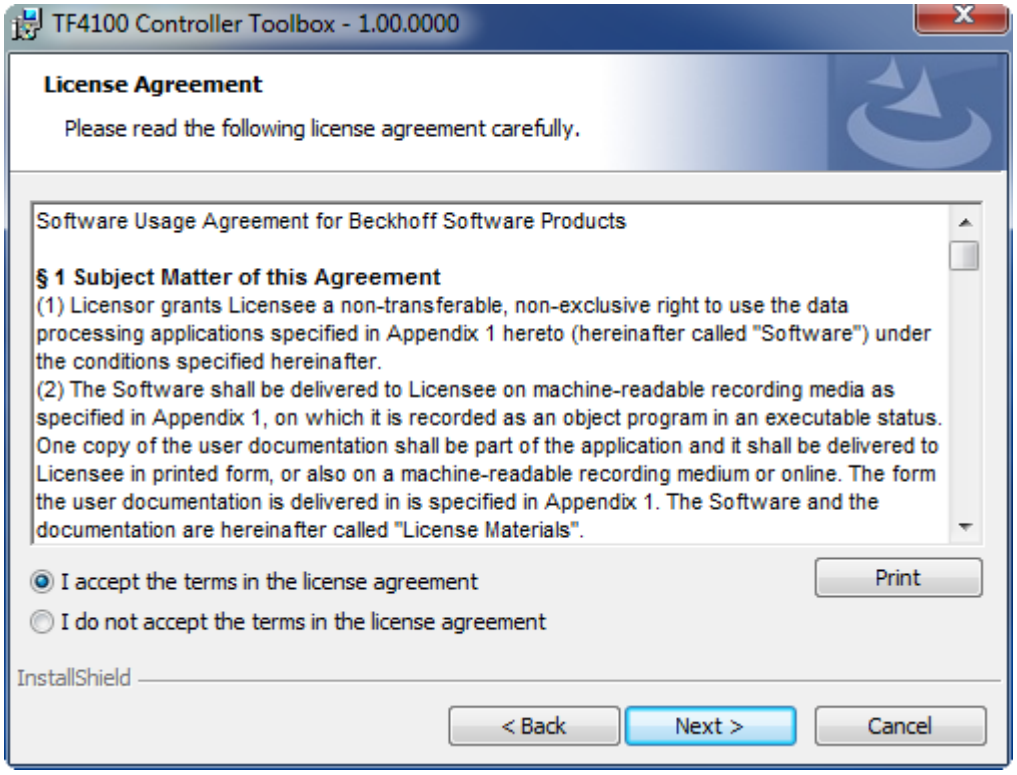
OPC-UA Sample Client

- Betriebssysteme:
  - Windows XP Pro SP3
  - Windows 7 Pro (32-bit und 64-bit)
  - Windows XP Embedded
  - Windows Embedded Standard 2009
  - Windows Embedded 7
  - Windows Server 2008 R2
  - Windows Server 2012
- Sonstiges:
  - .NET Framework 4.0

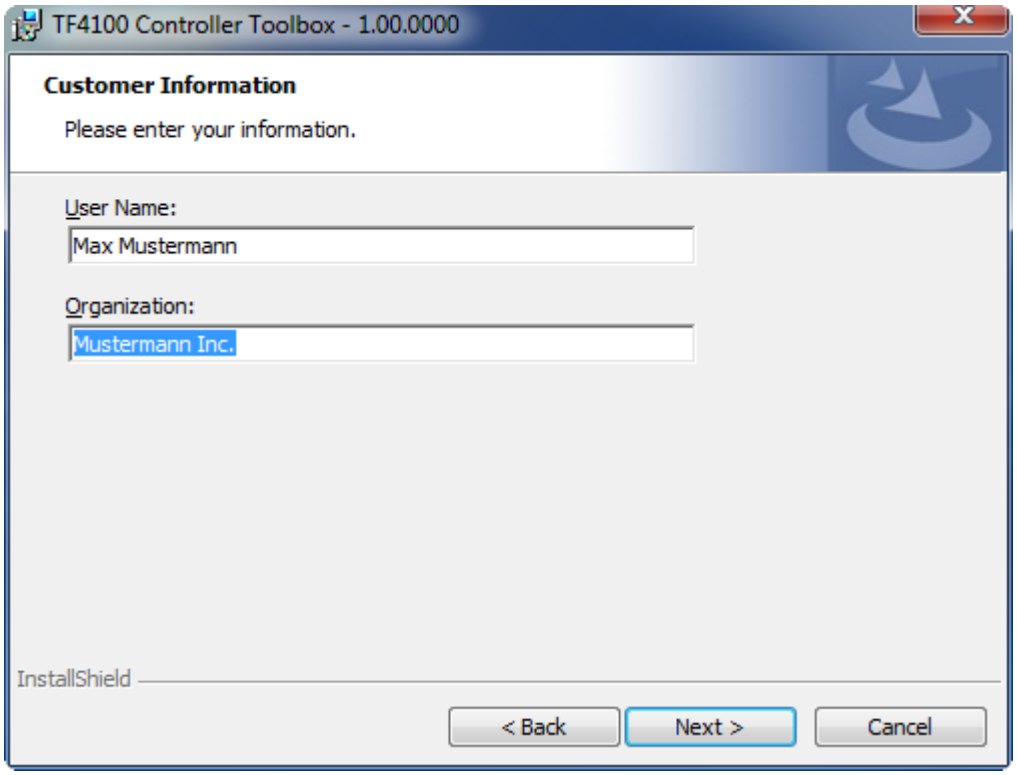
3.2 Installation

Die Installation der TwinCAT 3 Function für Windows basierte Betriebssysteme erfolgt Schritt-für-Schritt.

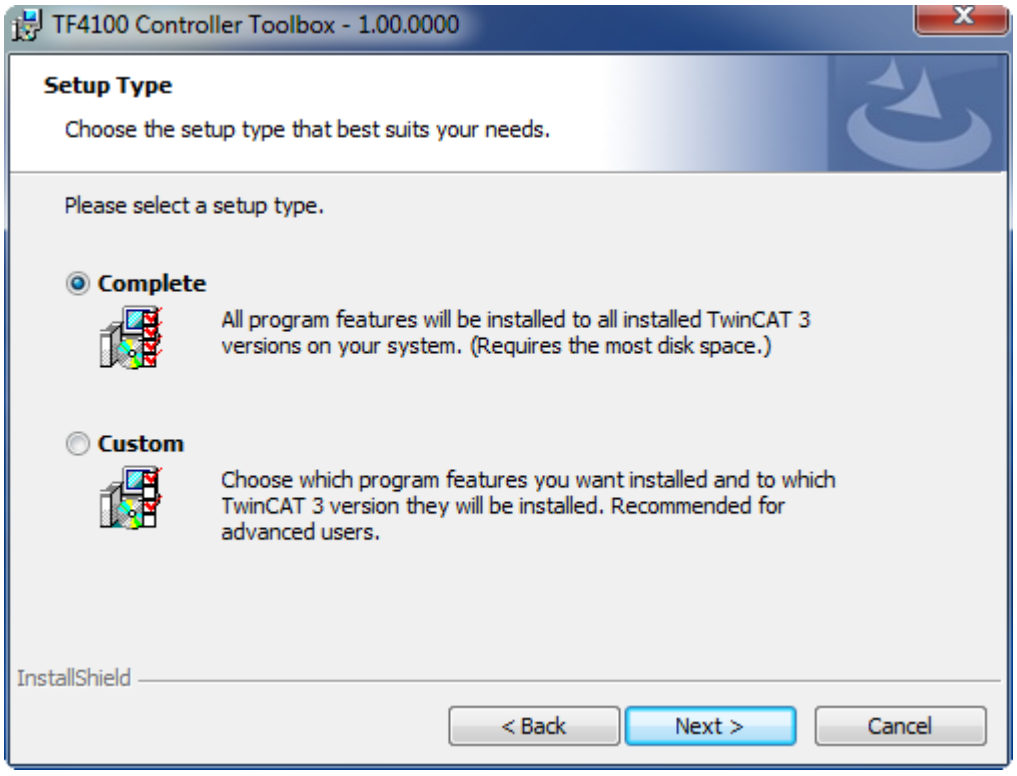
1. Führen Sie einen Doppelklick auf die herunter geladene Datei „TFxxxx" aus.
   Hinweis: Bitte starten Sie die Installation unter Windows per „Als Administrator ausführen", indem Sie die Setup-Dateien mit der rechten Maus anklicken und die entsprechende Option im Kontextmenü auswählen.
2. Klicken Sie auf „Next" und akzeptieren Sie dann die Endbenutzervereinbarung


TF4100 Controller Toolbox - 1.00.0000
License Agreement
Please read the following license agreement carefully.
Software Usage Agreement for Beckhoff Software Products
§ 1 Subject Matter of this Agreement
(1) Licensor grants Licensee a non-transferable, non-exclusive right to use the data processing applications specified in Appendix 1 hereto (hereinafter called "Software") under the conditions specified hereinafter.
(2) The Software shall be delivered to Licensee on machine-readable recording media as specified in Appendix 1, on which it is recorded as an object program in an executable status. One copy of the user documentation shall be part of the application and it shall be delivered to Licensee in printed form, or also on a machine-readable recording medium or online. The form the user documentation is delivered in is specified in Appendix 1. The Software and the documentation are hereinafter called "License Materials".
I accept the terms in the license agreement
I do not accept the terms in the license agreement
Print
InstallShield
< Back
Next >
Cancel

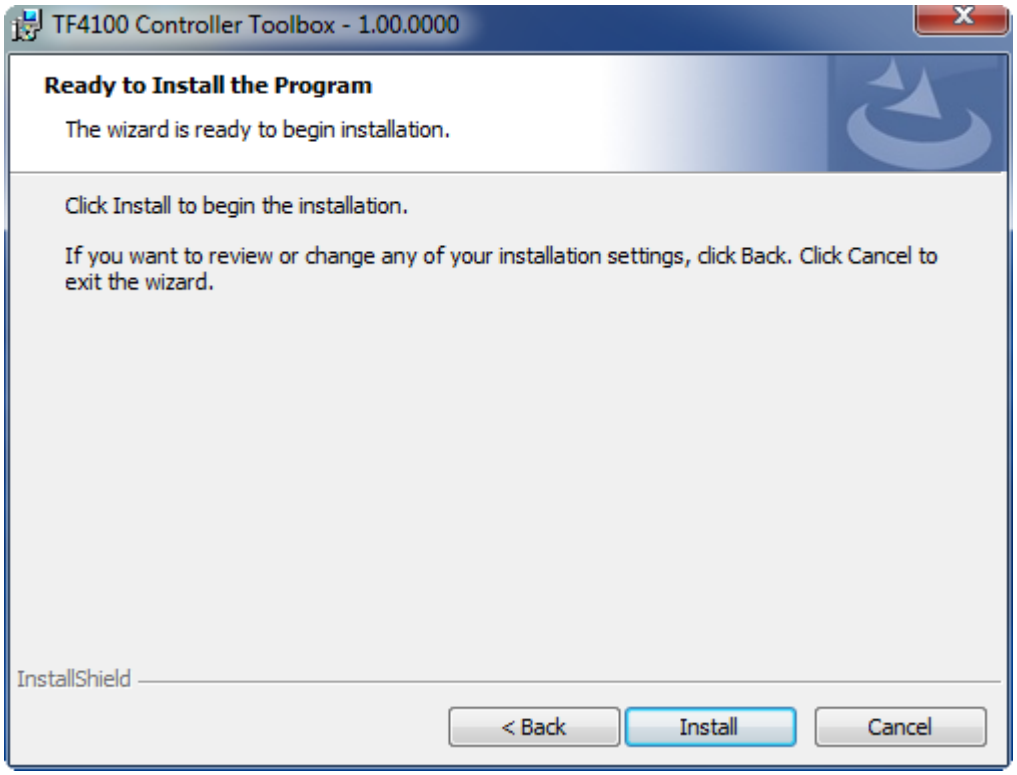
3. Geben Sie Ihre Benutzerdaten ein.



4. Für eine vollständige Installation wählen Sie **„Complete"** als Installationstyp. Alternativ können Sie jede Komponente separat installieren, indem Sie **"Custom"** wählen.

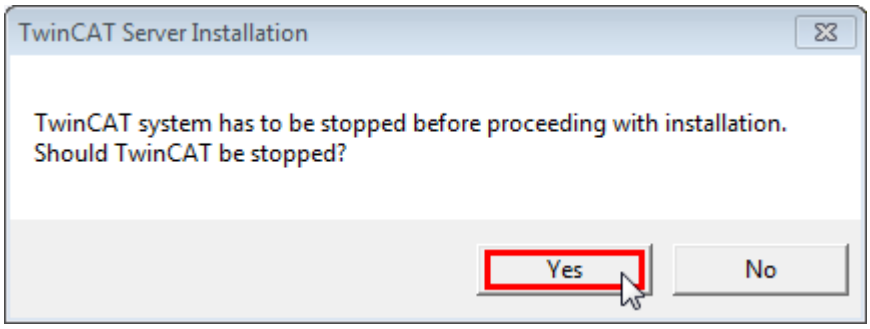

TF4100 Controller Toolbox - 1.00.0000
Setup Type
Choose the setup type that best suits your needs.
Please select a setup type.
Complete
All program features will be installed to all installed TwinCAT 3 versions on your system. (Requires the most disk space.)
Custom
Choose which program features you want installed and to which TwinCAT 3 version they will be installed. Recommended for advanced users.
InstallShield
< Back
Next >
Cancel

5. Wählen Sie **„Next“** und **„Install"** um die Installation zu beginnen.

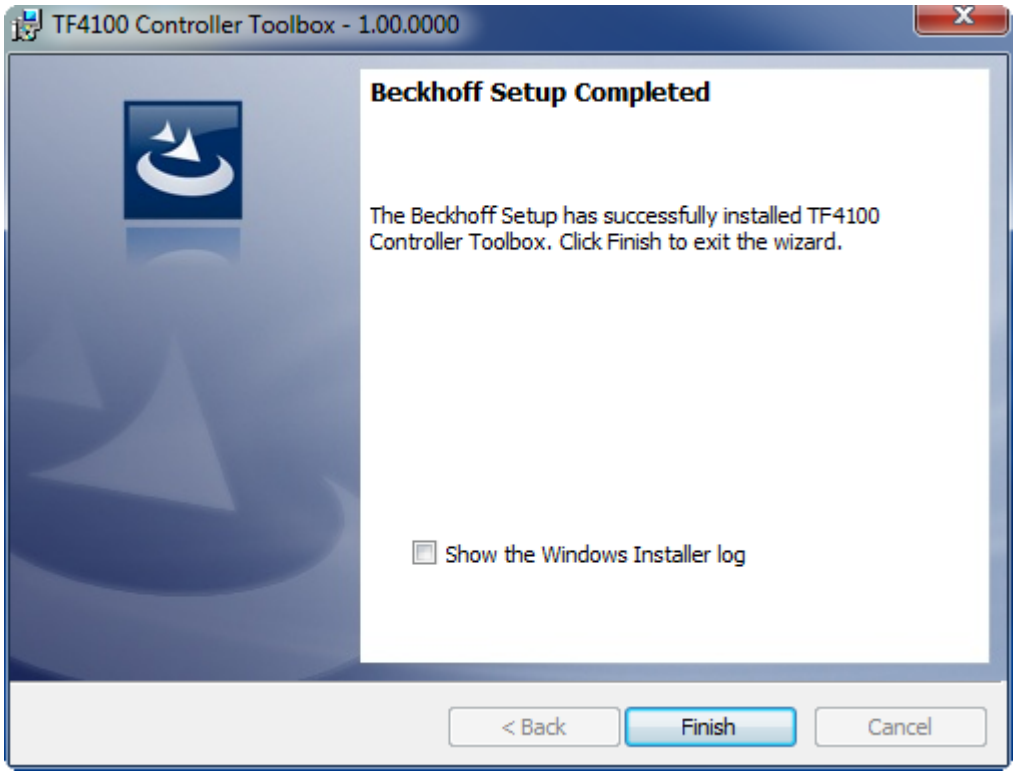

TF4100 Controller Toolbox - 1.00.0000
Ready to Install the Program
The wizard is ready to begin installation.
Click Install to begin the installation.
If you want to review or change any of your installation settings, click Back. Click Cancel to exit the wizard.
InstallShield
< Back
Install
Cancel

Das TwinCAT System muss gestoppt werden um mit der Installation fortzufahren.

6. Bestätigen Sie den Dialog mit **„Yes“**



7. Wählen Sie **„Finish"** um das Setup zu beenden.


TF4100 Controller Toolbox - 1.00.0000
Beckhoff Setup Completed
The Beckhoff Setup has successfully installed TF4100 Controller Toolbox. Click Finish to exit the wizard.
Show the Windows Installer log
< Back
Finish
Cancel

⇨ Damit ist die Installation abgeschlossen.

Der nächste Schritt nach einer erfolgreichen Installation ist die Lizensierung der TC3 Function [▶ 31].

3.3 Installation Windows CE

Dieser Teil der Dokumentation beschreibt, wie die TwinCAT 3 Function TF6100 OPC-UA auf einem Beckhoff Embedded Controller mit Windows CE installiert werden kann.

Der Setup Prozess besteht aus 4 Schritten:

- Download der Setup Datei
- Installation auf einem Host Computer
- Übertragung der ausführbaren Datei auf das Windows CE Gerät
- Installation der Software
- Upgrade der Software

Download der Setup Datei

Die ausführbare Datei für Windows CE ist Teil des TF6100 OPC-UA Setups. Daher müssen Sie nur das entsprechende Setup von www.beckhoff.com beziehen, welches automatisch alle Versionen für Windows XP, Windows 7 und Windows CE (x86 und ARM) enthält.

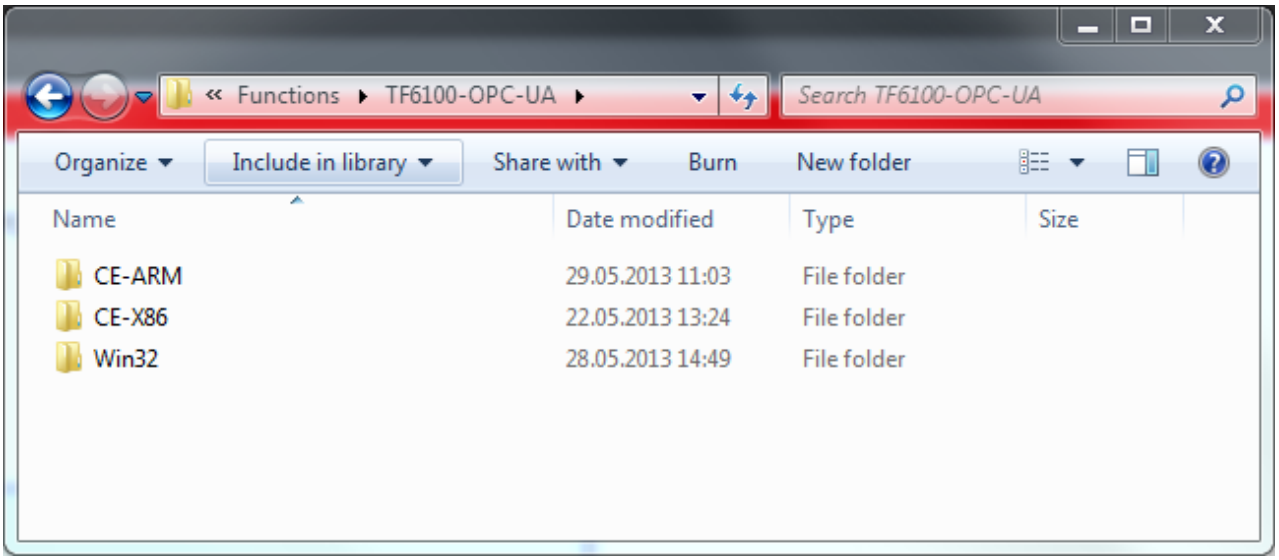
Die Installationsbeschreibung für das TF6100 OPC-UA Setup ist in unserem regulären Installationsartikel enthalten.

Installation auf einem Host Computer

Nach der Installation enthält der Installationsordner drei Verzeichnisse - jeweils ein Verzeichnis pro Hardwareplattform:

- **CE-ARM:** ARM-basierte Embedded Controller, welche unter Windows CE laufen, z.B. CX8090, CX9020

- **CE-X86:** X86-basierte Embedded Controller, welche unter Windows CE laufen, z.B. CX50xx. CX20x0
- **Win32:** Embedded Controller, welche unter Windows XP, Windows 7 oder Windows Embedded Standard laufen



Die Verzeichnisse CE-ARM und CE-X86 enthalten die CAB-Dateien Dateien der TF6100 Function für Windows CE - in Bezug auf die jeweilige Hardwareplattform Ihres Windows CE Geräts. Diese ausführbaren Dateien müssen auf das Windows CE Gerät kopiert werden. Wie dies funktioniert erfahren Sie im nächsten Kapitel.

Übertragung der ausführbaren Datei auf das Windows CE Gerät

Übertragen Sie die ausführbare Datei auf Ihr Windows CE Gerät. Zur Dateiübertragung stehen Ihnen mehrere Wege offen:

- über Netzwerkfreigaben
- über den integrierten FTP-Server
- über ActiveSync
- über CF/SD Karten

Für weitere Informationen konsultieren Sie bitte den Windows CE Bereich in unserem Information System.

Installation der Software

Nachdem die Datei auf das Windows CE Gerät übertragen wurde, müssen Sie die Datei dort ausführen. Den darauf folgenden Installationsdialog können Sie mit OK bestätigen. Nachdem die Installation beendet wurde, starten Sie das CE Gerät neu.

Nachdem das Gerät neu gestartet wurde, werden die ausführbaren Dateien (Client und Server) automatisch im Hintergrund geladen und sind nun verfügbar.

Die Software wird in dem folgenden Verzeichnis auf dem CE-Gerät installiert: **\Hard Disk\TwinCAT \Functions\TF6310-TCP-IP**

Upgrade der Software

Falls Sie schon eine ältere TF6100 Version auf dem Windows CE Gerät installiert haben, so müssen Sie die folgenden Schritte auf dem Windows CE Gerät durchführen, um auf eine neuere Version zu upgraden:

1. Öffnen Sie den CE Explorer, indem Sie auf Start --> Run klicken und "explorer" eingeben
2. Navigieren Sie nach \Hard Disk\TwinCAT\Functions\Tf6100-OPC-UA\Server bzw. (in einem zweiten Schritt) \Hard Disk\TwinCAT\Functions\Tf6100-OPC-UA\Client
3. Benennen Sie die Datei TcOpcUaServer.exe bzw. TcOpcUaClient.exe um

4. Starten Sie das Windows CE Gerät neu
5. Übertragen Sie die neue CAB-Datei auf das Windows CE Gerät
6. Installieren Sie die neue CAB-Datei
7. Löschen Sie die alten (umbenannten) Dateien
8. Starten Sie das Windows CE Gerät neu

⇨ Nachdem der Neustart durchgeführt wurde ist die neue Version aktiv.

3.4 Installation Tc2

In diesem Teil der Dokumentation finden Sie eine Schritt-für-Schritt Anleitung zum Setup von TwinCAT OPC-UA unter Windows XP. Folgende Punkte sind in diesem Dokument beschrieben:

- Download der Setup-Datei
- Starten der Installation
- Nach der Installation

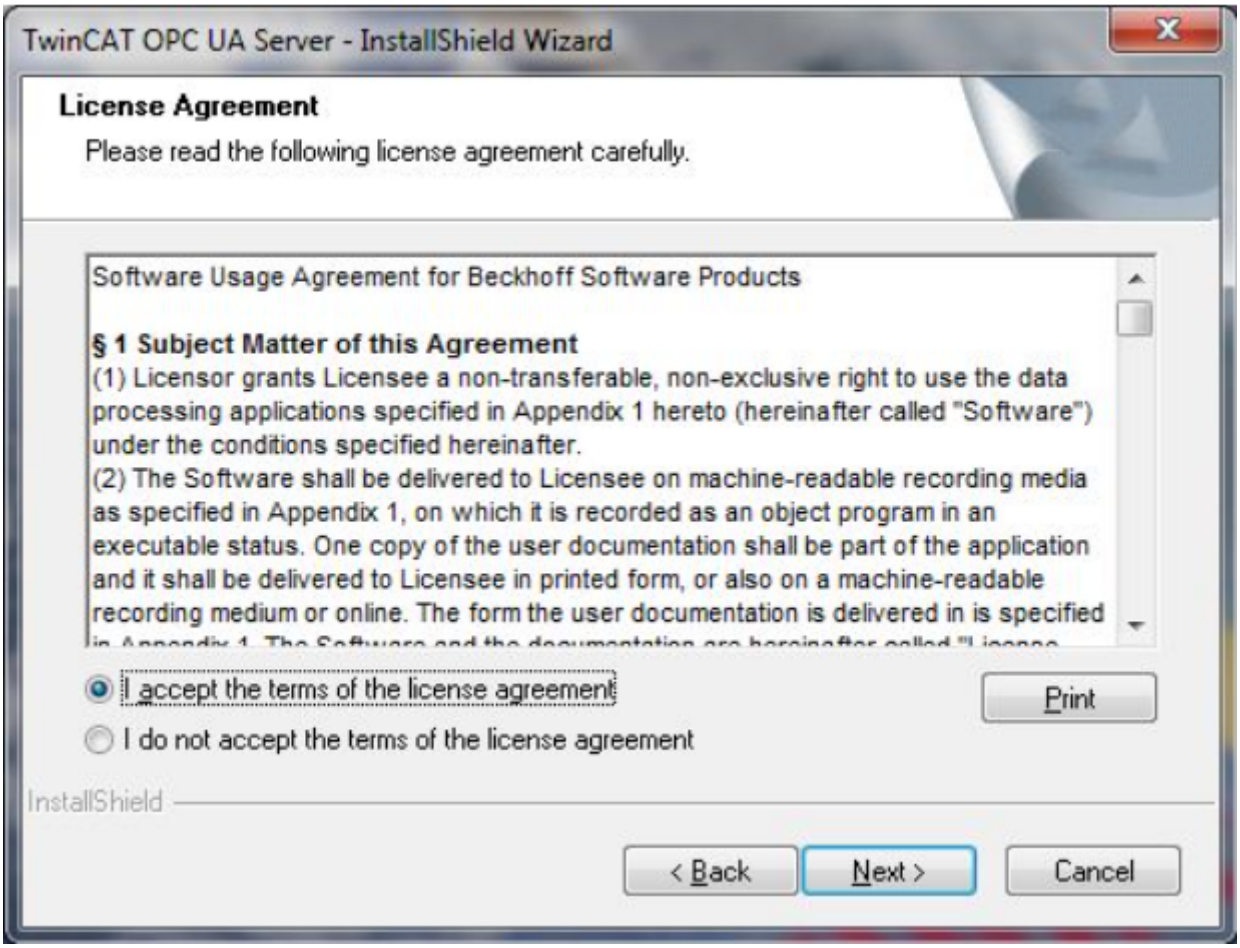
Download der Setup-Datei

Genau wie viele andere TwinCAT Ergänzungsprodukte steht auch OPC-UA als Download auf der Beckhoff Internetseite zur Verfügung. Die Download-Version ist die geläufigste. Sie ist als 30-Tage Demoversion oder Vollversion verfügbar.

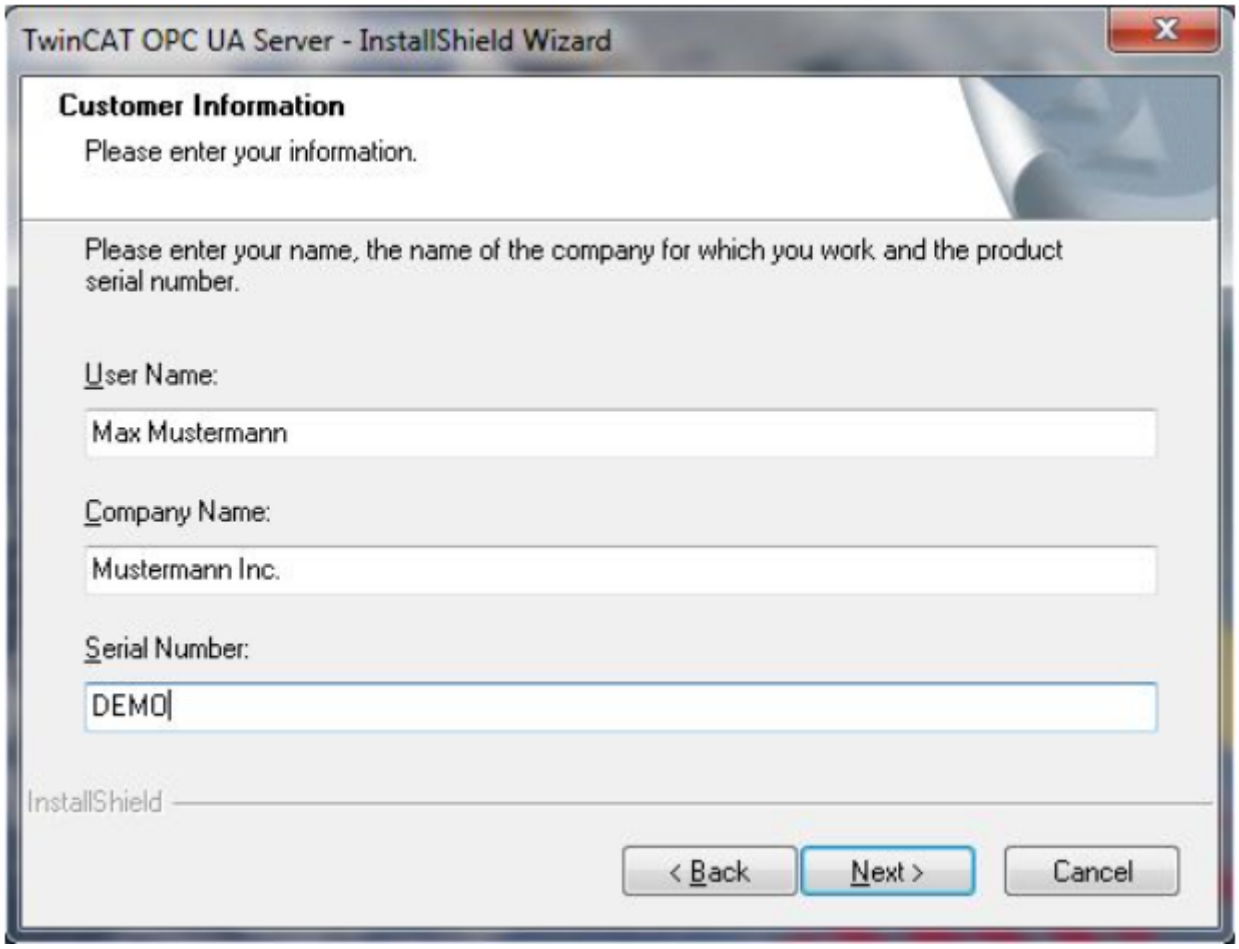
Starten der Installation

Zum Installieren des Supplements gehen Sie wie folgt vor:

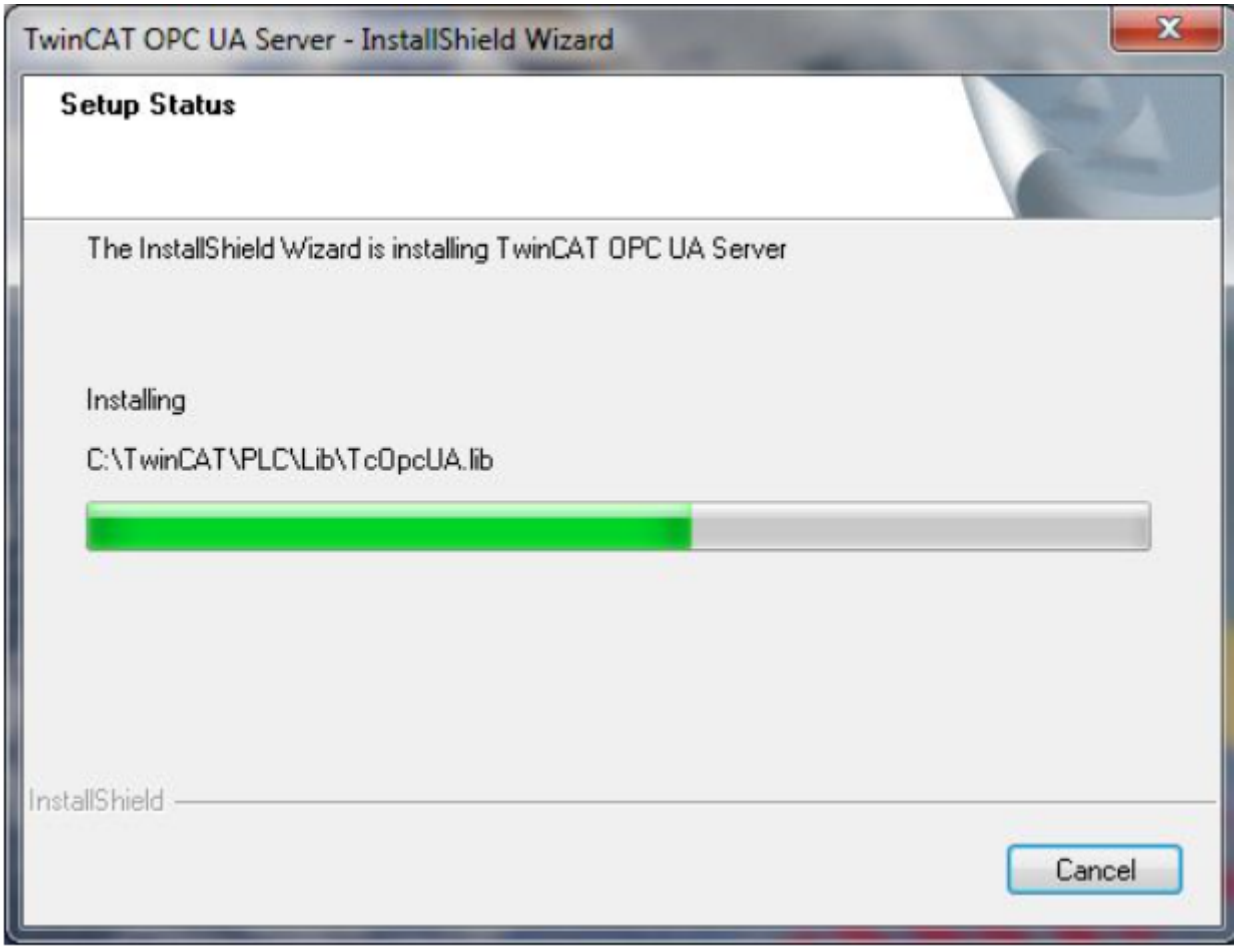
- Doppelklicken Sie auf die heruntergeladene Setupdatei TcOpcUaSvr.exe. **Bitte beachten:** Unter Windows 7 32-bit/64-bit starten Sie die Installation mit "Als Administrator ausführen" indem Sie mit der rechten Maustaste auf die Setup-Datei klicken und die entsprechende Option im Kontextmenü auswählen
- Wählen Sie die Installationssprache
- Klicken Sie auf „Weiter“ und stimmen der Lizenzvereinbarung zu


TwinCAT OPC UA Server - InstallShield Wizard
License Agreement
Please read the following license agreement carefully.
Software Usage Agreement for Beckhoff Software Products
§ 1 Subject Matter of this Agreement
(1) Licensor grants Licensee a non-transferable, non-exclusive right to use the data processing applications specified in Appendix 1 hereto (hereinafter called "Software") under the conditions specified hereinafter.
(2) The Software shall be delivered to Licensee on machine-readable recording media as specified in Appendix 1, on which it is recorded as an object program in an executable status. One copy of the user documentation shall be part of the application and it shall be delivered to Licensee in printed form, or also on a machine-readable recording medium or online. The form the user documentation is delivered in is specified
I accept the terms of the license agreement
I do not accept the terms of the license agreement
Print
InstallShield
< Back
Next >
Cancel

- Geben Sie ihre Benutzerinformationen ein. Alle Felder müssen ausgefüllt werden. Wenn Sie eine 30-Tage-Demo installieren möchten, geben Sie bitte „DEMO“ als Lizenzschlüssel ein.



- Klicken Sie auf „Installieren“, um die Installation zu starten.



- Nach dem Setup müssen Sie den Computer neu starten

Nach der Installation

Die TwinCAT Ergänzung OPC-UA wird automatisch beim Setup-Prozess konfiguriert und es sind keine weiteren Einstellungen für die Nutzung dieses Produkts erforderlich. Zu den weiteren Schritten gehören gegebenenfalls:

- Eine erste Verbindung mit dem installierten UA-Server herstellen und dessen Konfiguration mit der OPC-UA SampleClient Software testen.
- Das UA-Server Setup mit Hilfe des OPC-UA Configurator's personalisieren.
- Vergewissern Sie sich zudem, dass Ihre Firewall den TCP Port 4870 öffnet, weil der OPC-UC Server diesen Port benötigt.

3.5 Installation Windows CE Tc2

In diesem Teil der Dokumentation wird beschrieben, wie Sie die Ergänzung TwinCAT OPC-UA auf einer Windows CE basierten Beckhoff Embedded PC-Steuerung, z.B. CX1000, CX1020, CX9000, CX9001, CX9010, CX8090, CP62xx, C69xx, … installieren können.

- Der Setup-Prozess besteht aus vier Schritten:
- Download der Setup-Datei
- Installierung auf einen Host-Computer
- Übertragen der Setup-Datei auf ein Windows CE Gerät
- Ausführen des Setup auf dem Windows CE Gerät.

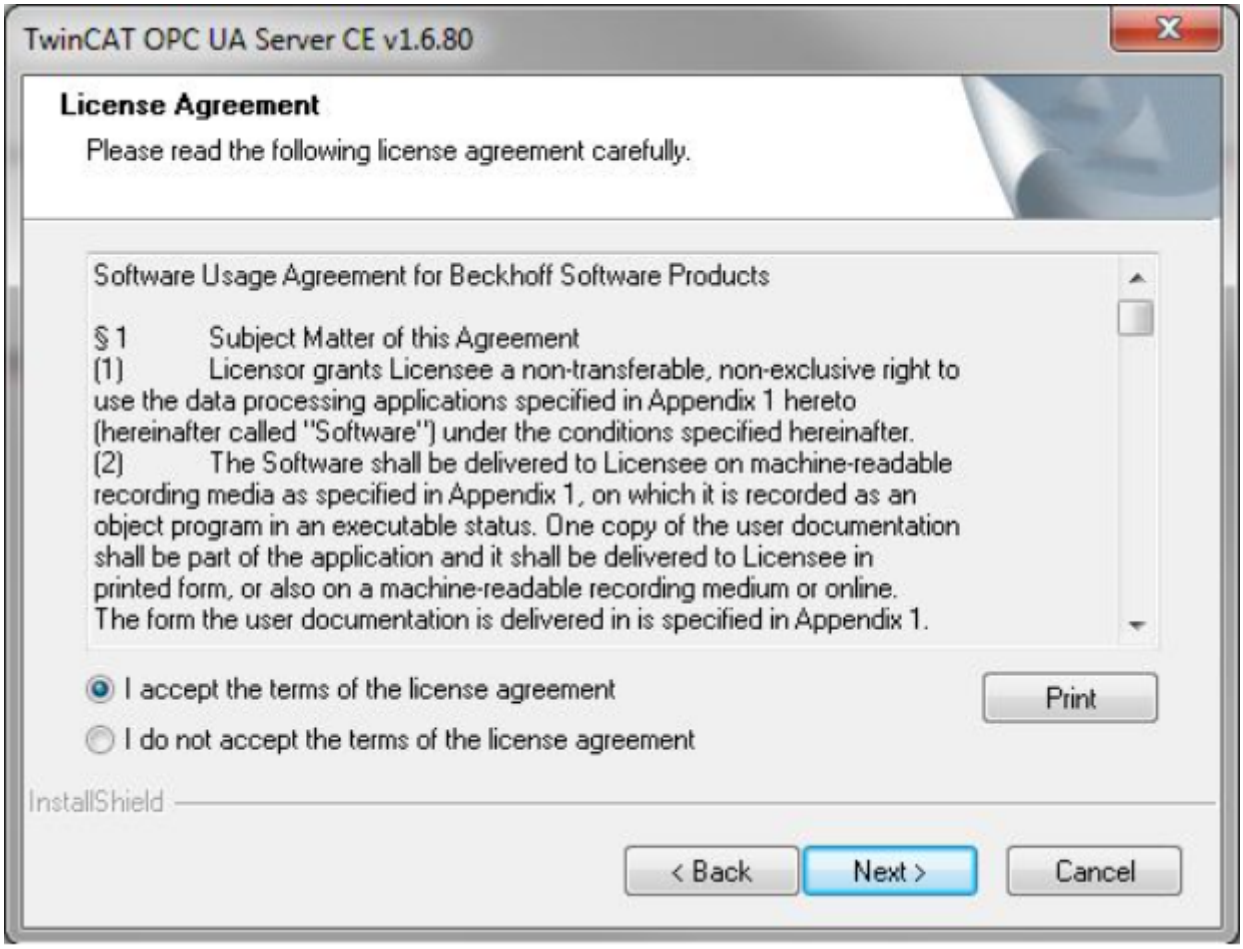
Download der Setup-Datei

So wie viele andere TwinCAT Ergänzungsprodukte steht auch OPC-UA für CE als Download auf dem Beckhoff FTP-Server zur Verfügung. Das Download stellt die aktuellste Version dar.

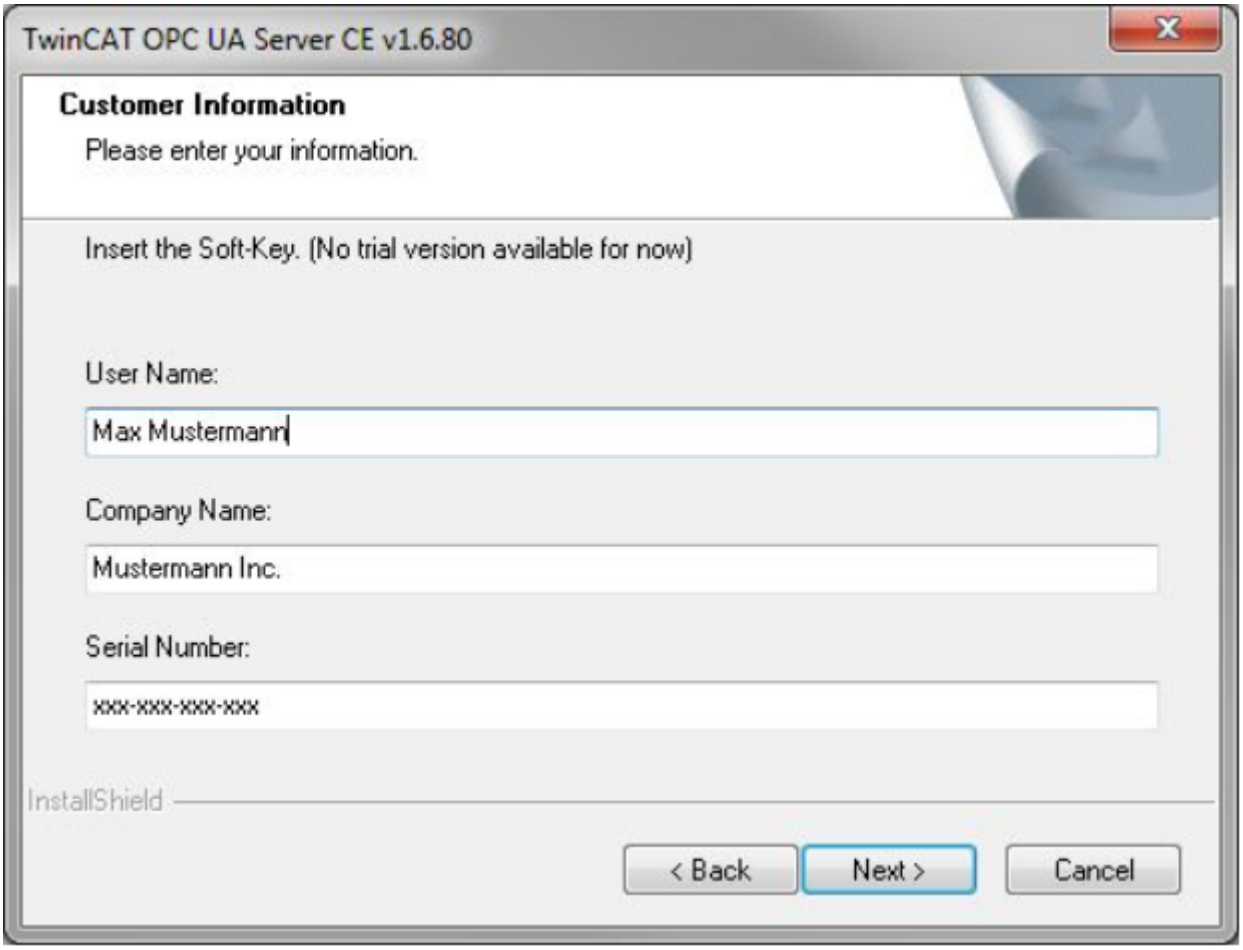
Installierung auf einen Host-Computer

Damit Sie die Installierungsdateien für Windows CE erhalten können, müssen Sie zunächst die herunter geladene Setup-Datei auf einem Host-Rechner, der ein beliebiges Windows CE basiertes System sein kann, installieren. Bitte führen Sie die folgenden Schritte aus:

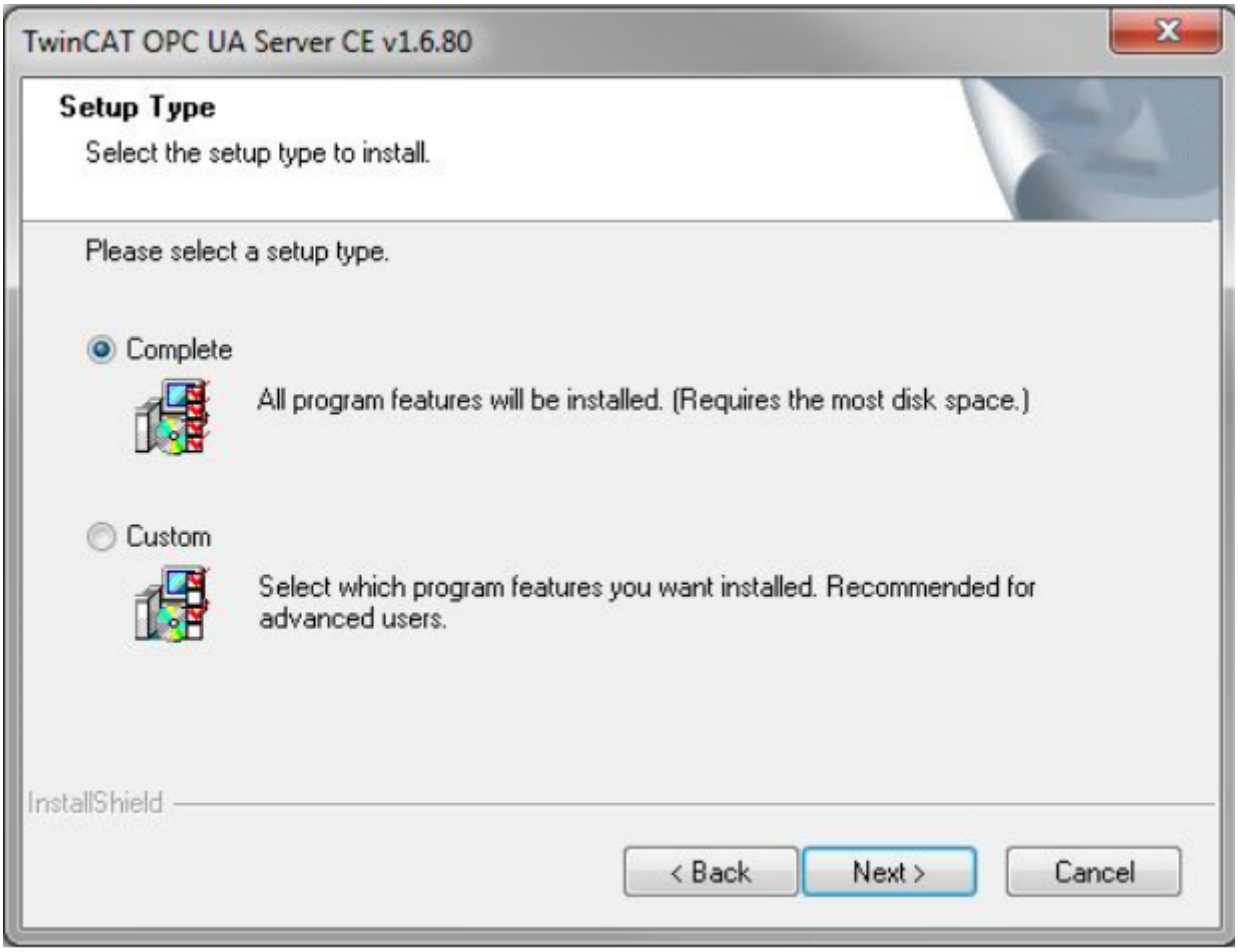
- Doppelklicken Sie auf die herunter geladene Datei TcOpcUaSvrCE.exe.
- Wählen Sie die Installations**sprache**
- Klicken Sie auf Weiter und stimmen der Lizenzvereinbarung zu


TwinCAT OPC UA Server CE v1.6.80
License Agreement
Please read the following license agreement carefully.
Software Usage Agreement for Beckhoff Software Products
§ 1 Subject Matter of this Agreement
(1) Licensor grants Licensee a non-transferable, non-exclusive right to use the data processing applications specified in Appendix 1 hereto (hereinafter called "Software") under the conditions specified hereinafter.
(2) The Software shall be delivered to Licensee on machine-readable recording media as specified in Appendix 1, on which it is recorded as an object program in an executable status. One copy of the user documentation shall be part of the application and it shall be delivered to Licensee in printed form, or also on a machine-readable recording medium or online.
The form the user documentation is delivered in is specified in Appendix 1.
I accept the terms of the license agreement
I do not accept the terms of the license agreement
Print
InstallShield
< Back
Next >
Cancel

- Geben Sie ihre **Benutzerinformationen** ein. Alle Felder müssen ausgefüllt werden. Beachten Sie, dass OPC-UA für CE derzeit nicht als Demo-Version verfügbar ist. Demzufolge benötigen Sie einen gültigen Lizenzschlüssel, um mit der Installierung fortzufahren.



- Wählen Sie als Installationsart „Vollständig“ und klicken Sie auf „Weiter“


TwinCAT OPC UA Server CE v1.6.80
Setup Type
Select the setup type to install.
Please select a setup type.
Complete
All program features will be installed. (Requires the most disk space.)
Custom
Select which program features you want installed. Recommended for advanced users.
InstallShield
< Back
Next >
Cancel

- Klicken Sie auf „Installieren“, um die Installation zu starten.

Nach der Installierung finden Sie die Setup-Dateien für Windows CE im Verzeichnis **".\TwinCAT\CE"**. Dieses Verzeichnis beinhaltet Setup-Dateien für die folgenden CE-Plattformen:

- TwinCAT OPC UA Client CE\I586: OPC-UA Client (SPS-Bibliothek) für x86 basierte CPUs (wie CX10xx, CP62xx, C69xx, ...)
- TwinCAT OPC UA Client CE\ARMV4I: OPC-UA Client (SPS-Bibliothek) für ARM basierte CPUs (wie CX9001, CX9010, CP6608, ...)
- TwinCAT OPC UA Server CE\I586: OPC-UA Server für x86 basierte CPUs (wie CX10xx, CP62xx, C69xx, ...)
- TwinCAT OPC UA Server CE\ARMV4I: OPC-UA Server für ARM basierte CPUs (wie CX9001, CX9010, CP6608, ...)

Übertragen der Setup-Datei auf ein Windows CE Gerät

Übertragen Sie die entsprechende Setup-Datei auf Ihr CE-Gerät. Dies kann wie folgt geschehen:

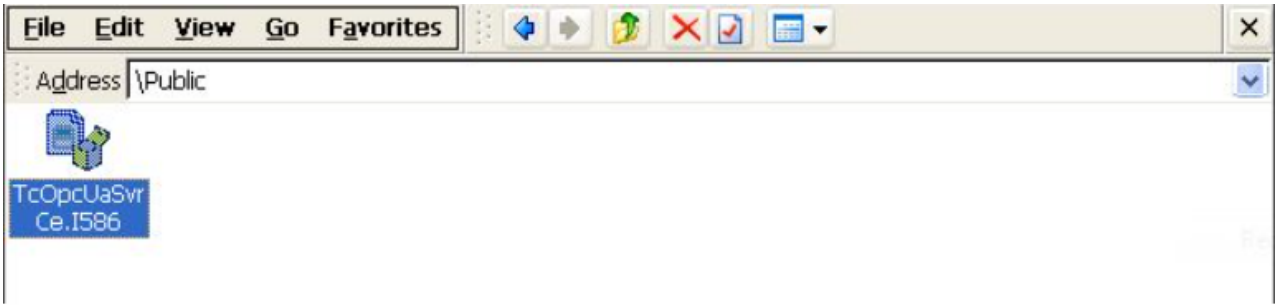
- über einen freigegebenen Ordner
- über den integrierten FTP-Server
- über ActiveSync
- über eine CF-Karte

Weitere Informationen finden Sie im „Windows CE“ Bereich in unserem Infosys Dokumentationssystem.

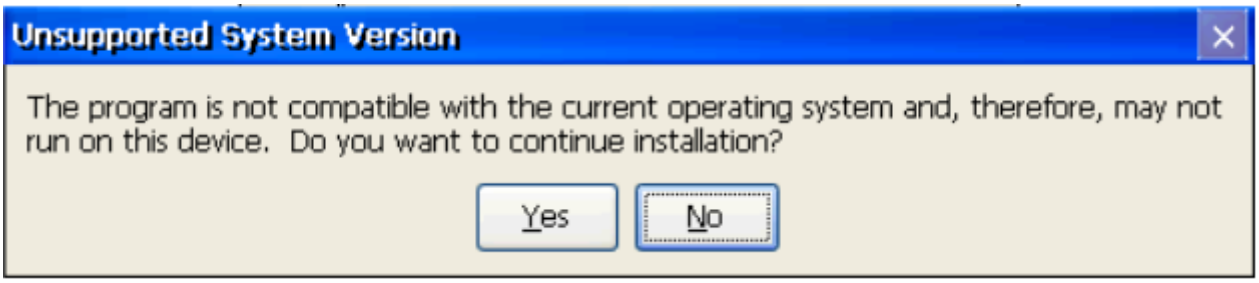
Ausführen des Setup auf dem Windows CE Gerät.

Nun muss die übertragene Setup-Datei TcOpcUaSvrCe.xxxx.CAB auf dem CE Gerät ausgeführt werden. Bitte führen Sie die folgenden Schritte aus:

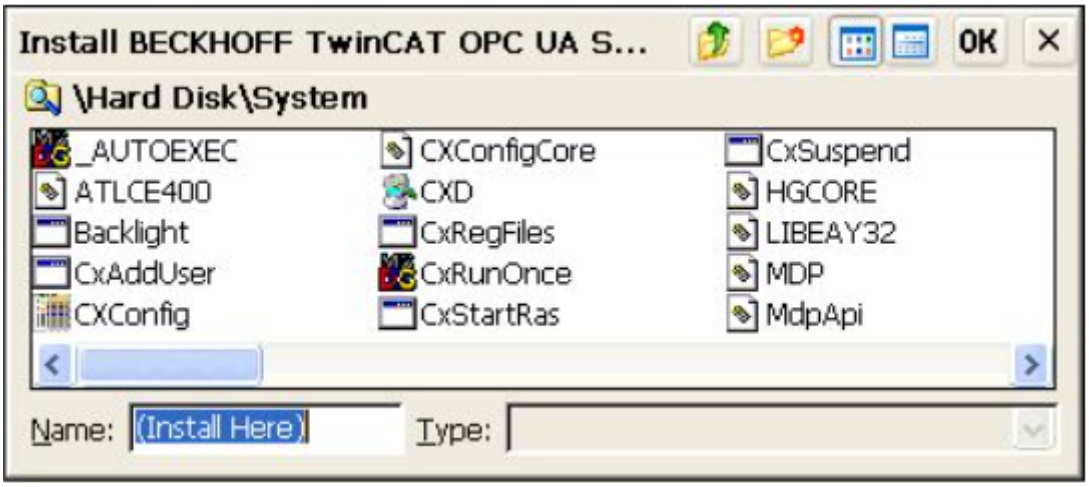
- Gehen Sie zum Verzeichnis, in das Sie die Setup-Datei übertragen haben.



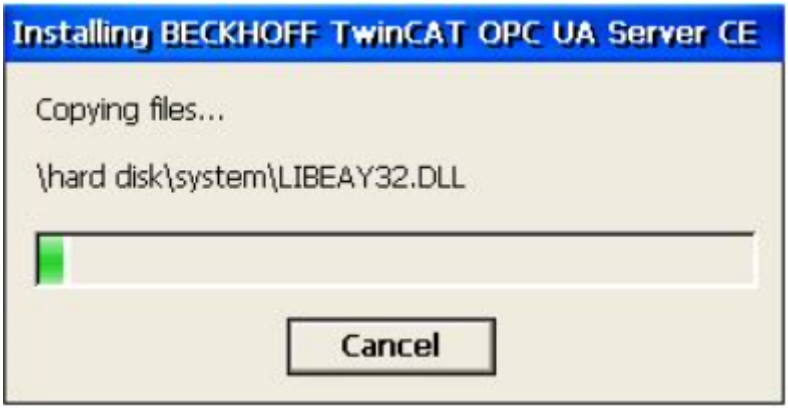
- Doppelklicken Sie auf die CAB-Datei. Wird ein Meldungsdialog eingeblendet, in dem behauptet wird, dass dieses Programm nicht mit dem aktuellen Betriebssystem kompatibel ist, müssen Sie sich erneut vergewissern, dass Sie die richtige CAB-Datei (ARM, I586) für Ihren IPC/Embedded-PC verwenden.
- Wenn Sie sicher sind, dass die CAB-Datei zum Embedded-PC/IPC passt, bestätigen Sie diesen Meldungsdialog mit „**Ja**“.


Unsupported System Version
The program is not compatible with the current operating system and, therefore, may not run on this device. Do you want to continue installation?
Yes
No

- Wählen Sie \Hard Disk\System\ als Zielverzeichnis



- Klicken Sie auf „OK“, um die Installation zu starten.



Nach der Installierung wird die Setup-Datei automatisch gelöscht. Beachten Sie, dass wenn Sie den OPC-UA Server installiert haben, diese Komponente nach einem Neustart Ihres CE-Geräts verfügbar wird.

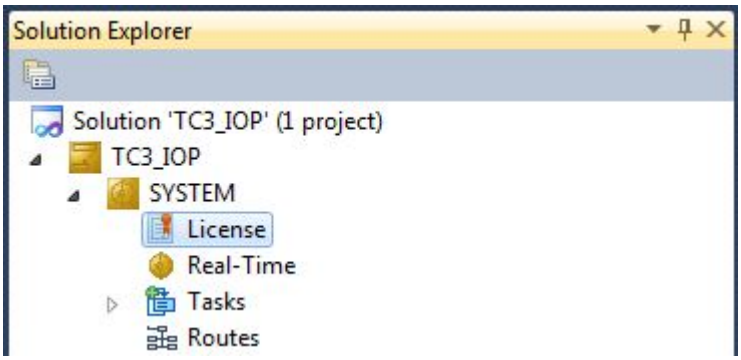
3.6 Lizensierung

Die TwinCAT 3 Function ist zusätzlich zur Vollversion auch in einer 7-Tage Testversion freischaltbar. Beide Lizenztypen sind über TwinCAT XAE aktivierbar. Weitere Information zum TwinCAT 3 Lizensierungsverfahren finden Sie im TwinCAT 3 Hilfesystem. Das folgende Dokument beschreibt den Lizensierungsvorgang einer TwinCAT 3 Function und gliedert sich dabei in die folgenden beiden Unterkapitel:

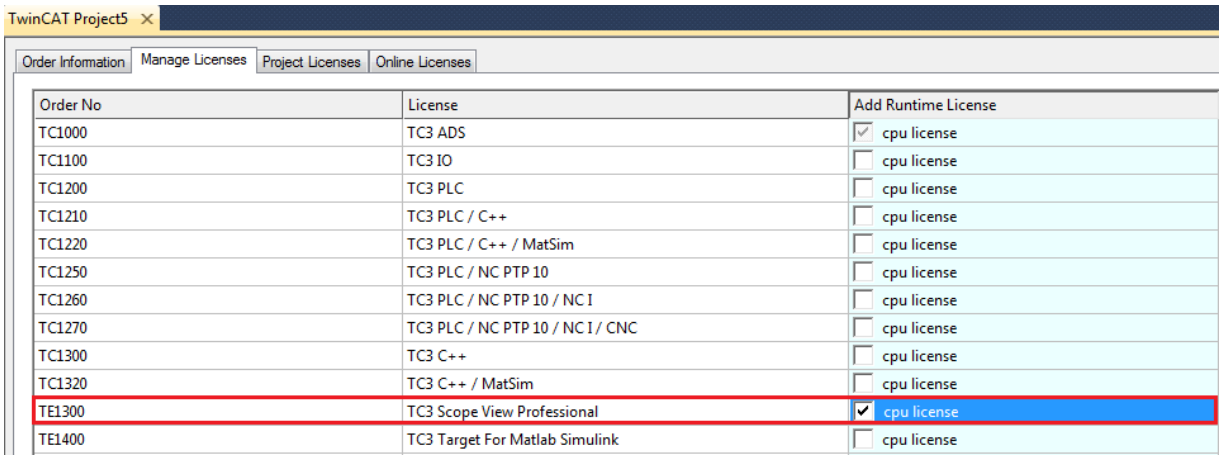
- Lizensierung einer 7-Tage Testversion [▶ 31]
- Lizensierung einer Vollversion [▶ 32]

Lizensierung einer 7-Tage Testversion

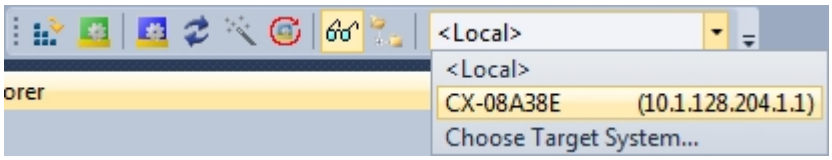
1. Starten Sie TwinCAT XAE
2. Öffnen Sie ein bestehendes TwinCAT 3 Projekt, oder legen Sie ein neues Projekt an
3. Navigieren Sie im “Solution Explorer” zum Eintrag **„System\License“**



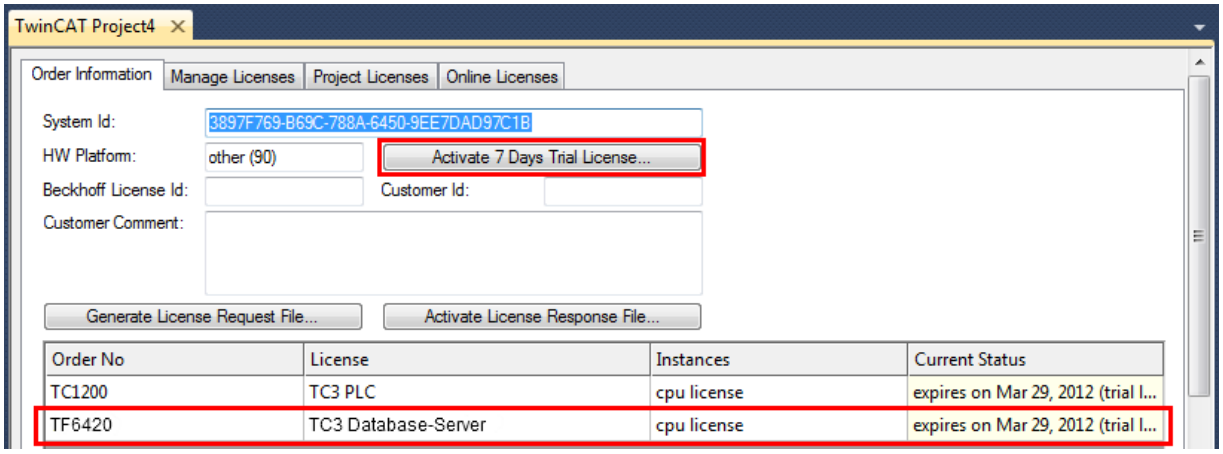
4. Öffnen Sie die Registerkarte **„Manage Licenses"** und fügen Sie eine **„Runtime License"** für Ihr Produkt hinzu (in diesem Screenshot „TE1300: TC3 Scope View Professional")


TwinCAT Project5
Order Information
Manage Licenses
Project Licenses
Online Licenses
Order No
License
Add Runtime License
TC1000 TC3 ADS cpu license
TC1100 TC3 IO cpu license
TC1200 TC3 PLC cpu license
TC1210 TC3 PLC / C++ cpu license
TC1220 TC3 PLC / C++ / MatSim cpu license
TC1250 TC3 PLC / NC PTP 10 cpu license
TC1260 TC3 PLC / NC PTP 10 / NC I cpu license
TC1270 TC3 PLC / NC PTP 10 / NC I / CNC cpu license
TC1300 TC3 C++ cpu license
TC1320 TC3 C++ / MatSim cpu license
TE1300 TC3 Scope View Professional cpu license
TE1400 TC3 Target For Matlab Simulink cpu license

5. **Optional** : Möchten Sie die Lizenz für ein Remote Gerät hinzufügen, müssen Sie sich zunächst mit diesem Gerät über die TwinCAT XAE Toolbar verbinden



6. Aktivieren Sie in der Registerkarte **„Order Information"** über den Button **„Activate 7 Days Trial License..."** eine Testversion

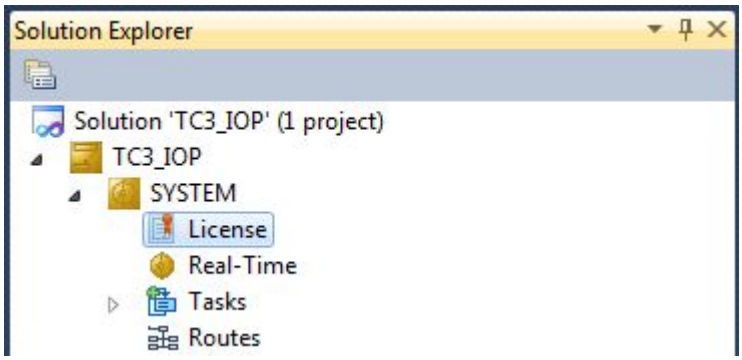


7. Starten Sie im Anschluss daran das TwinCAT 3 System einmal neu

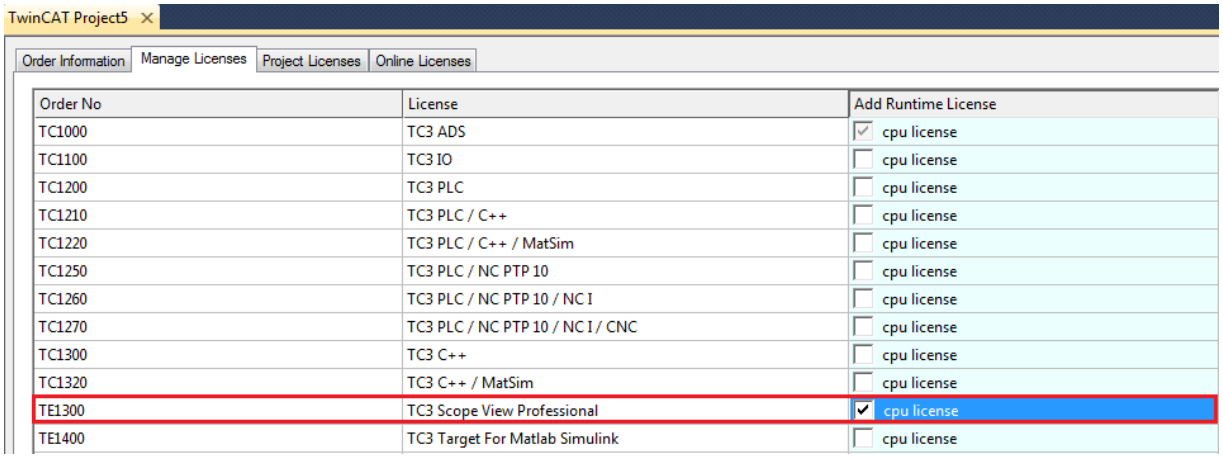
Lizensierung einer Vollversion

8. Starten Sie TwinCAT XAE
9. Öffnen Sie ein bestehendes TwinCAT 3 Projekt oder legen Sie ein neues Projekt an

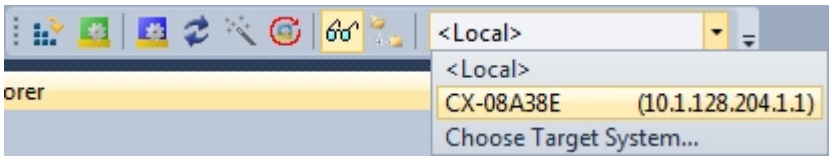
10. Navigieren Sie im "Solution Explorer" zum Eintrag **„SYSTEM/License"**



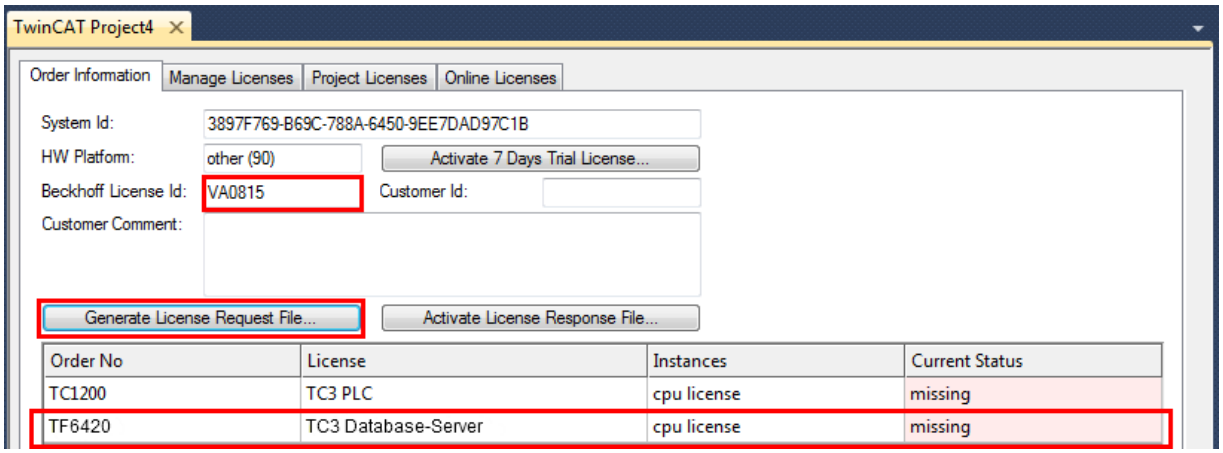
11. Öffnen Sie die Registerkarte **„Manage Licenses"** und fügen Sie eine **„Runtime License"** für Ihr Produkt hinzu (in diesem Screenshot " TE1300: TC3 Scope View Professional ").


TwinCAT Project5
Order Information
Manage Licenses
Project Licenses
Online Licenses
Order No | License | Add Runtime License
TC1000 | TC3 ADS | cpu license
TC1100 | TC3 IO | cpu license
TC1200 | TC3 PLC | cpu license
TC1210 | TC3 PLC / C++ | cpu license
TC1220 | TC3 PLC / C++ / MatSim | cpu license
TC1250 | TC3 PLC / NC PTP 10 | cpu license
TC1260 | TC3 PLC / NC PTP 10 / NC I | cpu license
TC1270 | TC3 PLC / NC PTP 10 / NC I / CNC | cpu license
TC1300 | TC3 C++ | cpu license
TC1320 | TC3 C++ / MatSim | cpu license
TE1300 | TC3 Scope View Professional | cpu license
TE1400 | TC3 Target For Matlab Simulink | cpu license

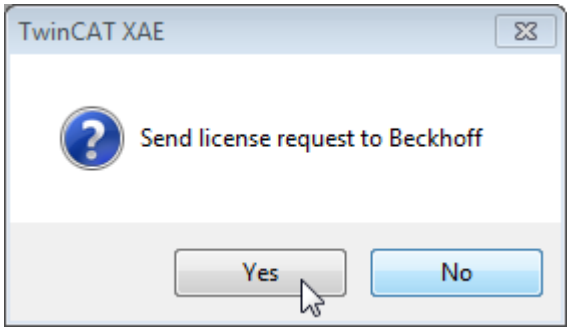
12. **Optional:** Möchten Sie die Lizenz für ein Remote Gerät hinzufügen, müssen Sie sich zunächst mit diesem Gerät über die TwinCAT XAE Toolbar verbinden



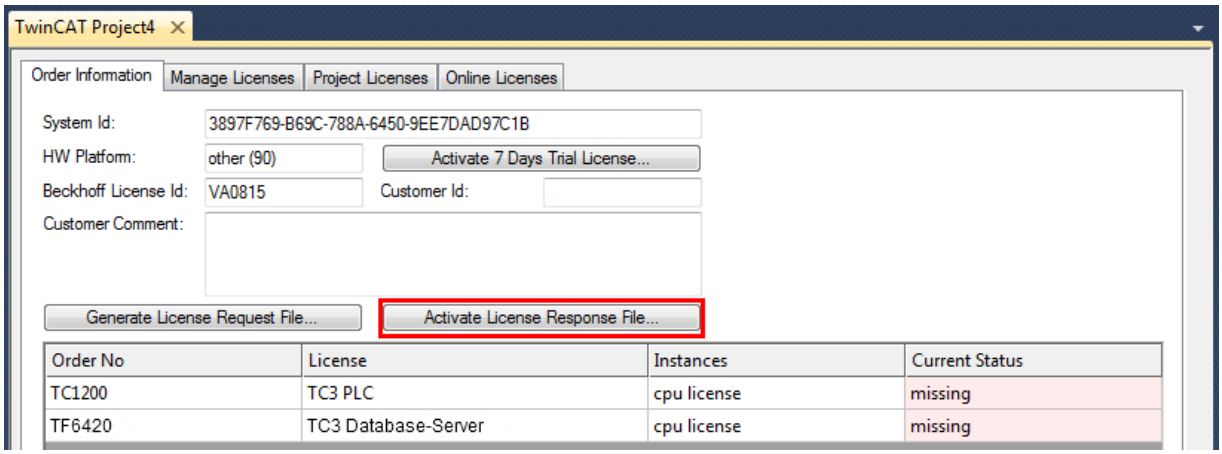
13. Öffnen Sie die Registerkarte **„Order Information"**
Die Felder „System-ID" und „HW Platform" können nicht geändert werden, sie beschreiben die zu lizensierende Plattform. Generell wird eine TwinCAT 3 Lizenz an zwei Kennzahlen gebunden:
Die „System-ID" identifiziert Ihr Gerät eindeutig.
Die „HW Platform" ist eine Kennzahl für die Performanz des Gerätes.
14. Tragen Sie optional eine eigene Bestellnummer und einen Kommentar für Ihre Zwecke ein


TwinCAT Project4
Order Information
Manage Licenses
Project Licenses
Online Licenses
System Id: 3897F769-B69C-788A-6450-9EE7DAD97C1B
HW Platform: other (90)
Activate 7 Days Trial License...
Beckhoff License Id: VA0815
Customer Id:
Customer Comment:
Generate License Request File...
Activate License Response File...
Order No | License | Instances | Current Status
TC1200 | TC3 PLC | cpu license | missing
TF6420 | TC3 Database-Server | cpu license | missing

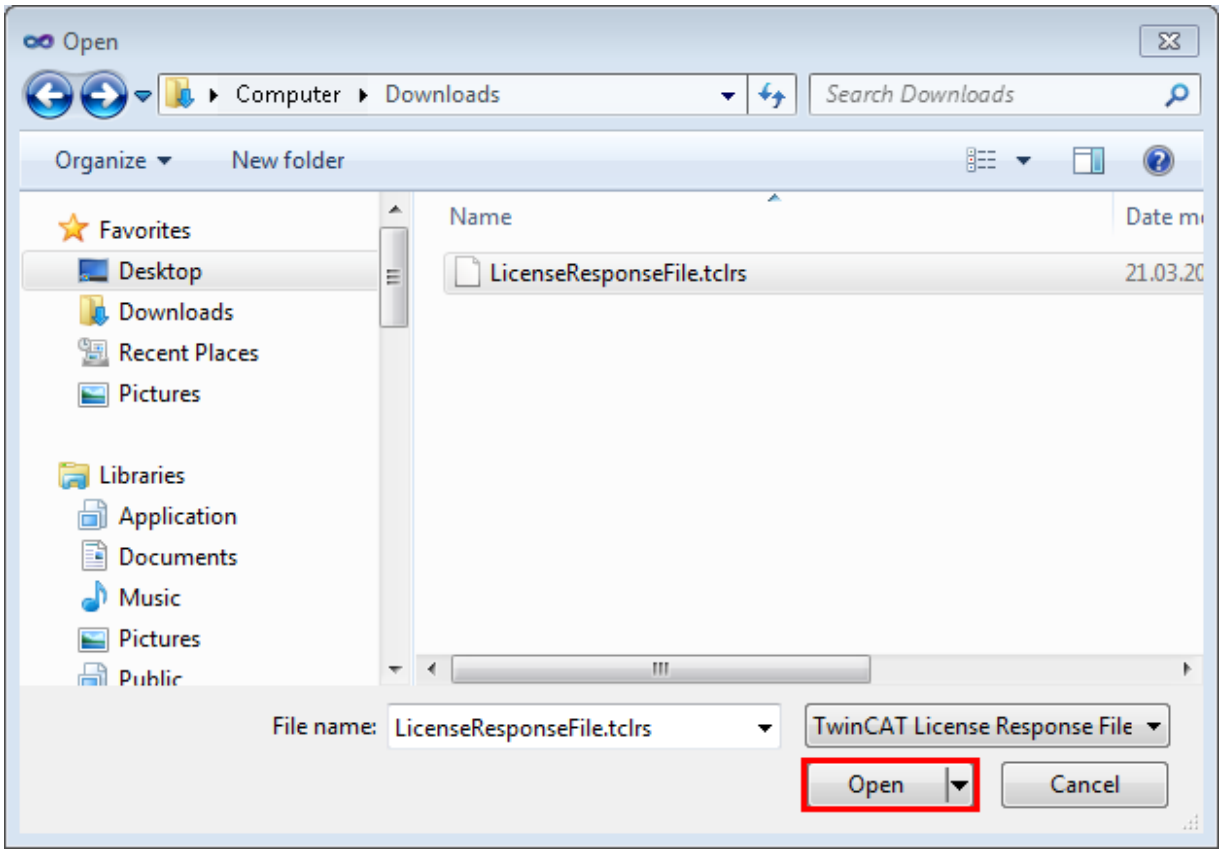
15. Generieren Sie in der Registerkarte **„Order Information"** über den Button **„Generate License Request File..."** eine Lizenzanforderungs-Datei, die durch einen Beckhoff-Lizenzserver validiert wird (wenn Ihnen Ihre **„Beckhoff License ID“** nicht bekannt ist, wenden Sie sich an Ihren Ansprechpartner aus dem Beckhoff Vertrieb).
16. Nachdem Sie das „License Request File“ gespeichert haben, fragt das System, ob die Datei per Mail an den Beckhoff Lizenz Server geschickt werden soll:



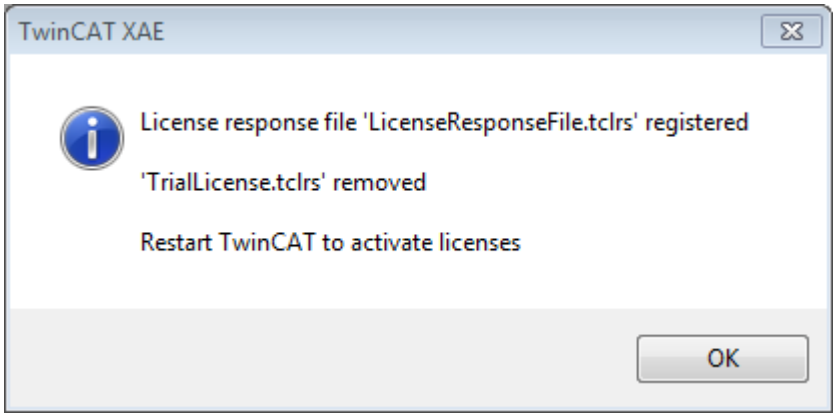
17. Wenn Sie den Dialog mit „Yes" bestätigen, öffnet sich Ihr Standard E-Mail Client und erzeugt eine neue E-Mail für „tclicense@beckhoff.com", die das „License Request File" enthält
18. Senden Sie diesen Activation Request an Beckhoff
    HINWEIS! das „License Response File“ wird an die dieselbe E-Mail Adresse versendet, die das „License Request File“ verschickt hat
19. Kurz darauf erhalten Sie vom Beckhoff-Lizenzserver eine Lizenzdatei,
    importieren Sie sie über den Button „Activate License Response File...“, um das Produkt zu aktivieren



20. Wählen Sie in Ihrem Ordnersystem das erhaltene „License Response File" aus



21. Das „License Response File" wird importiert und alle enthaltenen Lizenzen werden aktiviert, sämtliche betroffenen Demo-Lizenzen werden entfernt
22. Starten Sie TwinCAT neu, um die Lizenz zu aktivieren



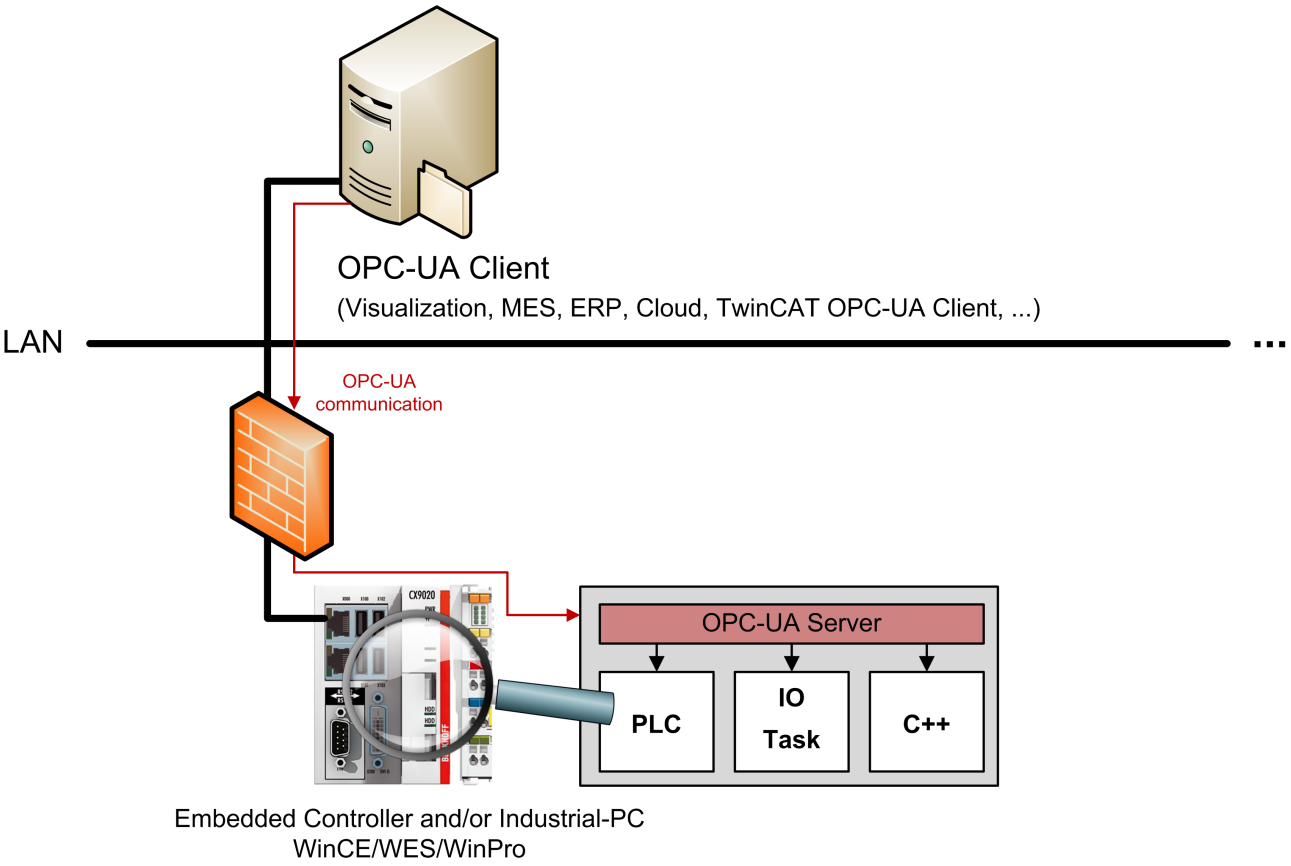
HINWEIS! Das Lizenzfile wird automatisch auf Ihre lokale Festplatte unter „...\TwinCAT \3.1\Target\License" kopiert.

4 Konfiguration

4.1 Server

4.1.1 Übersicht

Der TwinCAT OPC-UA Server bietet eine standardisierte Kommunikationsschnittstelle für den Zugriff auf Symbolwerte aus der TwinCAT SPS- und C++-Laufzeit bzw. I/O-Task.



Dieser Teil der Dokumentation beschreibt die Basis-Konfiguartion des TwinCAT OPC-UA Servers.

Hinweis: Einen guten Einstiegspunkt in die Thematik bietet unsere Quick Start Anleitung.

Im Folgenden finden Sie Dokumentationen über diese Themenbereiche:

- OPC-UA Konfigurator: Grafische Oberfläche zur Konfiguration des OPC-UA Servers. Editiert dessen Konfigurationsdatei (siehe unten)
- OPC-UA DA (Data Access): Zugriff auf TwinCAT PLC- und IO-Task-Werte via OPC-UA
- OPC-UA HA (Historical Access): Bietet Möglichkeiten zur Speicherung von historischen Daten, z.B. im lokalen RAM, einer Datei oder einer Datenbank
- OPC-UA AC (Alarm & Conditioning): Bietet Condition Monitoring Funktionalitäten für SPS-Variablen
- OPC-UA Security: Beschreibt die in OPC-UA integrierten Sicherheitsmechanismen

Erweiterte Konfiguration

Gegebenenfalls sind, je nach Art Ihrer Betriebsumgebung, zusätzliche Konfigurationsparameter erforderlich, um die folgenden Dinge einzustellen:

- Konfigurationsdatei [▶ 78]: Beschreibung der Konfigurationsdatei "ServerConfig.xml" des OPC-UA Servers. Diese Datei sollte überlicherweise über den OPC-UA Konfigurator editiert werden (siehe oben)
- Konfiguration von SPS Variablen [▶ 59]: Bietet eine Übersicht über alle konfigurierbaren Optionen (über SPS-Kommentare) an einer SPS-Variablen
- Weitere ADS-Geräte [▶ 79]: Beschreibt wie Sie weitere TwinCAT Geräte in den Namensraum des OPC-UA Servers integrieren
- Mehrere UA-Server [▶ 81]: Beschreibt wie Sie mehrere OPC-UA Server Instanzen auf einem Gerät starten können
- UA über NAT [▶ 83] : Beschreibt wie Sie eine OPC-UA Kommunikation über ein NAT-Gerät herstellen, z.B. eine Firewall oder einen Router
- Ändern der Handhabung von Arrays im OPC-UA Namensraum [▶ 58]: Beschreibt zwei Darstellungsoptionen von SPS-Arrays im OPC-UA Namensraum

4.1.2 Quick Start

Nach der erfolgreichen Installation und Lizensierung können die folgenden Schritte durchgeführt werden, um eine erste Verbindung zu einer SPS-Laufzeit über den OPC-UA-Server aufzubauen:

- Schritt 1: Konfiguration von SPS-Variablen für den OPC-UA Zugriff
- Schritt 2: Konfiguration des OPC-UA Namensraums
- Schritt 3: Herstellen einer Verbindung zum OPC-UA Server

Dieser Quick Start Artikel geht davon aus, dass Sie den OPC-UA Server auf demselben System wie die SPS-Laufzeit installiert haben.

Schritt 1: Konfiguration von SPS-Variablen für den OPC-UA Zugriff

Ein bestehendes SPS-Projekt öffnen und den folgenden Kommentar **vor** den ausgewählten Variablen einfügen:

TwinCAT 3 (mit TMC Import):

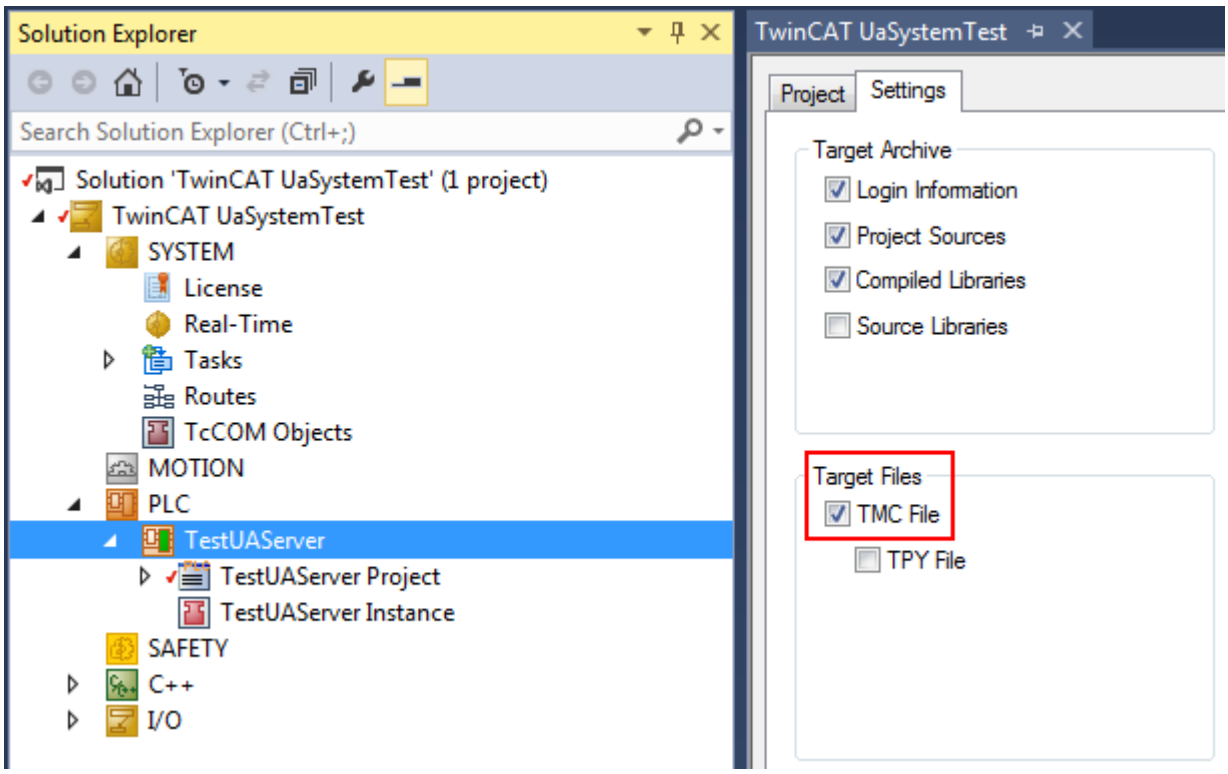
```
{attribute 'OPC.UA.DA' := '1'}
bVariable : BOOL;
```

TwinCAT 2 (mit TPY Import):

```
bVariable : BOOL; (*~ (OPC:1:some description) *)
```

Schritt 2: Den OPC-UA Namensraum konfigurieren

Der OPC-UA Server stellt standardmäßig eine Verbindung mit der ersten SPS-Laufzeit auf dem lokalen System her (lokal = das gleiche System, auf dem auch der OPC-UA Server installiert ist) und verwendet dessen entsprechende Symboldatei (TMC-Datei) für die Konfiguration des OPC-UA Namensraums. Damit diese Symboldatei automatisch an die SPS-Laufzeit übergeben wird, muss der Download der Symboldatei im SPS-Projekt aktiviert sein.



Bei jeder Aktivierung des SPS-Projekts wird die Symboldatei automatisch in das Verzeichnis C:\TwinCAT\3.x \Boot\Plc\ kopiert und dort nach dem entsprechenden ADS-Port des SPS-Projekts benannt, z.B. "Port_851.tmc" für die erste SPS-Laufzeit.

Die Standardkonfiguration des OPC-UA Servers liest diese Symboldatei automatisch beim Start.

Jetzt die Konfiguration aktivieren und TwinCAT neustarten. In SPS-Laufzeit einloggen und das SPS-Programm starten.

Mit Schritt 3 fortfahren.

Schritt 3: Verbindung mit OPC-UA Server herstellen

Um eine Verbindung von einem OPC-UA Client herzustellen, muss der Client eine Verbindung mit der URL des OPC-UA Servers herstellen. Diese URL wird folgendermaßen beschrieben:

opc.xxx://<ip or name>:port

- opc = vordefiniert
- xxx = Transport an OPC-UA Server, z. B. "tcp"
- port = Portnummer von OPC-UA Server, z. B. 4840 (standardmäßig)

Hinweis	**Wichtiger Hinweis** Beachten Sie bitte, dass der Standard-Port 4840 eventuell von anderen OPC-UA Servern verwendet wird, z. B. dem Local Discovery Server (LDS) von der OPC Foundation, die von manchen Anbietern mit OPC-UA Softwarepaketen eingesetzt wird.

Beispiele:

- opc.tcp://CX_0215AF:4840
- opc.tcp://172.16.2.80:4840

Starten Sie den mitgelieferten UA Sample Client (bei Windows-Systemen mit aktiviertem UAC, z. B. Windows 7, mit der Option „Run As Administrator“) und stellen eine Verbindung mit der URL des OPC-UA Servers her, z. B. opc.tcp://localhost:4840, mittels Klicken auf „Endpunkte erhalten“ und anschließendem „Verbinden“.

HINWEIS! Sie finden den UA Sample Client unter dem Windows Start Menü oder im Installationsverzeichnis des Produkts, z. B. C:\TwinCAT\Functions\TF6100-OPC-UA\Win32\SampleClient\.

Nach erfolgreicher Herstellung der Verbindung finden Sie die SPS-Laufzeit „PLC1“ mit den veröffentlichten Variablen (siehe Schritt 1) im UA-Namensraum unterhalb des Knotens „Objects“.

4.1.3 SPS

4.1.3.1 Zugriff auf SPS Laufzeit

In diesem Teil der Dokumentation wird beschrieben, wie der Namensraum des TwinCAT OPC-UA Servers konfiguriert werden kann. Der UA-Namensraum beschreibt die Struktur, in welche die SPS-Variablen abgebildet werden. Damit eine SPS-Variable über den UA-Namensraum erreichbar ist, muss sie dafür freigegeben werden. Der folgende Leitfaden beschreibt mehrere Möglichkeiten, um Variablen in einem SPS-Programm freizugeben.

Bitte beachten: Zu diesem Zeitpunkt sollte noch einmal daran erinnert werden, dass der OPC-UA Server immer standardmäßig mit dem ersten lokalen SPS-Laufzeitsystem in Verbindung tritt, dass also eine Konfiguration in den meisten Fällen nicht erforderlich ist. Siehe auch hierzu Artikel bezüglich Quick Start. Eine separate Konfiguration ist nur in außergewöhnlichen Fällen notwendig, z.B. wenn weitere Laufzeiten im UA-Namensraum anzuzeigen sind, oder wenn auf die SPS eines BC-Geräts zugegriffen werden muss.

Dieser Artikel beinhaltet folgende Themen:

- Allgemeine Informationen
- Schritt 1: SPS-Variablen auswählen, die über OPC-UA öffentlich zugänglich werden sollen.
- Schritt 2: Herunterladen von Symbolbeschreibung in SPS-Projekt aktivieren
- [optionaler] Schritt 3: Datenzugriffgerät in OPC-UA Server konfigurieren

Allgemeine Informationen

Für die Konfiguration des OPC-UA Namensraums und den Zugriff auf SPS-Variablen stehen mehrere Parameter zur Verfügung, die sich auf Art und Weise, wie die SPS-Laufzeit angezeigt wird und wie die Kommunikation stattfindet, auswirken. Diese Parameter können bequem mit Hilfe des OPC-UA Konfigurators festgelegt werden. Alternativ dazu können Sie diese auch direkt in der ServerConfig.xml Konfigurationsdatei festlegen. Die folgende Tabelle gibt einen Überblick über alle Parameter, die beim Zugriff auf eine SPS-Laufzeit von Bedeutung sind.

Parameter	Beschreibung	Mögliche Werte
ADS Port	Definiert den ADS-Port, unter dem die SPS-Laufzeit zugänglich ist. Der ADS-Port kann in den Eigenschaften des SPS-Projekts gelesen werden.	800 (BC Controller) 801 (TwinCAT 2) 811 (TwinCAT 2) … 851 (TwinCAT 3 - standardmäßig) 852 (TwinCAT 3) …
AutoCfg	Definiert zunächst den Typ der für die Kommunikation verwendeten Ziel-Laufzeit, z.B. SPS, C++, I/O. Anschließend kann eine weitergehende Unterscheidung innerhalb dieser Kategorien stattfinden. Jede AutoCfg-Option steht sowohl als ungefilterte, als auch gefilterte Option zur Verfügung. Gefiltert bedeutet, dass der Nutzer bestimmen kann, welche SPS-Symbole über OPC-UA via SPS-Kommentaren veröffentlicht werden sollten (siehe unten). Bei Verwendung einer ungefilterten Option wird jedes SPS-Symbol an OPC-UA veröffentlicht.	7 TwinCAT 2 (TPY) 8 TwinCAT 2 (TPY) gefiltert 4040 TwinCAT 3 (TMC) 4041 TwinCAT 3 (TMC) gefiltert
AutoCfgSymFile	Symboldatei der jeweiligen SPS-Laufzeit. Standardmäßig wird die automatisch erzeugte Symboldatei der ersten SPS-Laufzeit des lokalen Systems importiert.	Pfad zur Symboldatei. Zeigt standardmäßig auf die Symboldatei (TMC) der ersten lokalen SPS-Laufzeit.
IoMode	Definiert die Methode für den Zugriff auf Symbole. Dies ist besonders wichtig für den Zugriff auf BC-Geräte.	1 (Zugriff über Handle - standardmäßig) 3 (Zugriff auf BC Controller)
ArraySubItemLegacySupport	Unterelemente eines Array werden standardmäßig nicht als separate Knoten im UA-Namensraum abgebildet. Stattdessen wird lediglich das Array als ein einzelnes Element abgebildet. Nichtsdestotrotz können UA-Clients über deren sogenannten "IndexRange" auf Unterelemente zugreifen. Unglücklicherweise unterstützen einige ältere OPC-UA Clients diese Möglichkeit noch nicht. Dieses Flag wurde eingeführt, sodass der Zugriff nichtsdestotrotz für diese Clients möglich wird. Das Flag sorgt dafür, dass jede Array-Position als separater Knoten im UA-Namensraum angezeigt wird. Achtung: Dies führt zu einem höheren Speicherbedarf des OPC-UA Servers!	0 (deaktiviert - standardmäßig) 1 (aktiviert)
Disabled	Deaktiviert die SPS-Laufzeit im UA-Namensraum, woraufhin der entsprechende Knoten nicht angezeigt wird. Wir empfehlen die Aktivierung dieses Parameters wenn bestimmte SPS-Laufzeiten zum Zeitpunkt der Projektplanung noch nicht verfügbar sind, weil z.B. die entsprechenden Geräte noch nicht ans Netzwerk angeschlossen sind.	0 (deaktiviert - standardmäßig) 1 (aktiviert)

In den folgenden Schritten wird anhand eines Beispiels gezeigt, wie die Variablen aus einem SPS-Programm in den UA-Namensraum importiert werden können und welche Schritte hierzu notwendig sind. Es wird hierbei vorausgesetzt, dass der OPC-UA Server, in seiner Standardkonfiguration (Lieferzustand nach Installierung), und die SPS-Laufzeit sich auf dem gleichen Computer befinden.

Hinweis	**Bitte beachten** Die in diesem Artikel enthaltenen Informationen, einschließlich die folgenden SPS-Kommentare/Pragmas basieren auf TwinCAT 3 und müssen demzufolge mit den AutoCfg Optionen “4041 TwinCAT 3 (TMC) gefiltert”, welches die Standardkonfiguration darstellt, verwendet werden. Bitte beachten Sie ebenfalls, dass die AutoCfg Optionen für TwinCAT 2 ein anderes SPS Kommentarformat verwenden (*~ (OPC:1:description) *), das nicht mehr in TwinCAT 3 verwendet wird, aber weiterhin aus Gründen der Kompatibilität unter Verwendung der AutoCfg Option “8” verfügbar ist.

Schritt 1: SPS-Variablen auswählen, die über OPC-UA öffentlich zugänglich werden sollen.

Der OPC-UA Server stellt automatisch eine Verbindung zur ersten SPS-Laufzeit auf dem lokalen System her. Die für OPC-UA im SPS-Programm (TwinCAT SPS-Steuerung) markierten SPS-Symbole werden bereitgestellt. Die Markierung geschieht über einen Kommentar an der geeigneten Stelle (Instanz, Struktur, Variable) im SPS-Programmcode. Dies kann am besten anhand von zwei Beispielen erläutert werden.

Beispiel 1:
In diesem Beispiel sind die SPS-Variablen bMemFlag1, bMemFlag2, bMemAlarm2 und iReadOnly über OPC-UA freigegeben. Die SPS-Variable “bMemAlarm1” sollte *nicht* über den OPC-UA Server zugänglich sein. Das SPS-Programm in PLC-Control würde dann so aussehen – hängt von der von Ihnen verwendeten TwinCAT Version ab.

TwinCAT 3 (mit TMC Import):

```
{attribute 'OPC.UA.DA' := '1'}
bMemFlag1 : BOOL;

{attribute 'OPC.UA.DA' := '1'}
bMemFlag2 : BOOL;

bMemAlarm1 : BOOL;

{attribute 'OPC.UA.DA' := '1'}
bMemAlarm2 : BOOL;

{attribute 'OPC.UA.DA' := '1'}
{attribute 'OPC.UA.DA.Access' := '1'}
iReadOnly    : INT;
```

TwinCAT 2 (mit TPY Import):

```
bMemFlag1 : BOOL; (*~ (OPC:1:some description) *)

bMemFlag2 : BOOL; (*~ (OPC:1:some description) *)

bMemAlarm1 : BOOL;

bMemAlarm2 : BOOL; (*~ (OPC:1:some description) *)

iReadOnly    : INT; (*~ (OPC:1:some description)
(OPC_PROP[0005]:1:read-only flag) *)
```

Beachten Sie, dass die TPY Formatierung auch in TwinCAT 3 SPS-Projekten für Migrationszwecke verwendet werden kann, wenn z.B. ein TwinCAT 2 Projekt bequem in ein TwinCAT 3 Projekt konvertiert werden soll.

Beachten Sie auch, dass aufgrund des zusätzlichen Kommentars “OPC.UA.DA.Access”, die Zugriffsebene für die Variable iReadOnly auf “ReadOnly” eingestellt wurde. Siehe verschiedene Möglichkeiten in der vollständigen Liste der SPS-Kommentare.

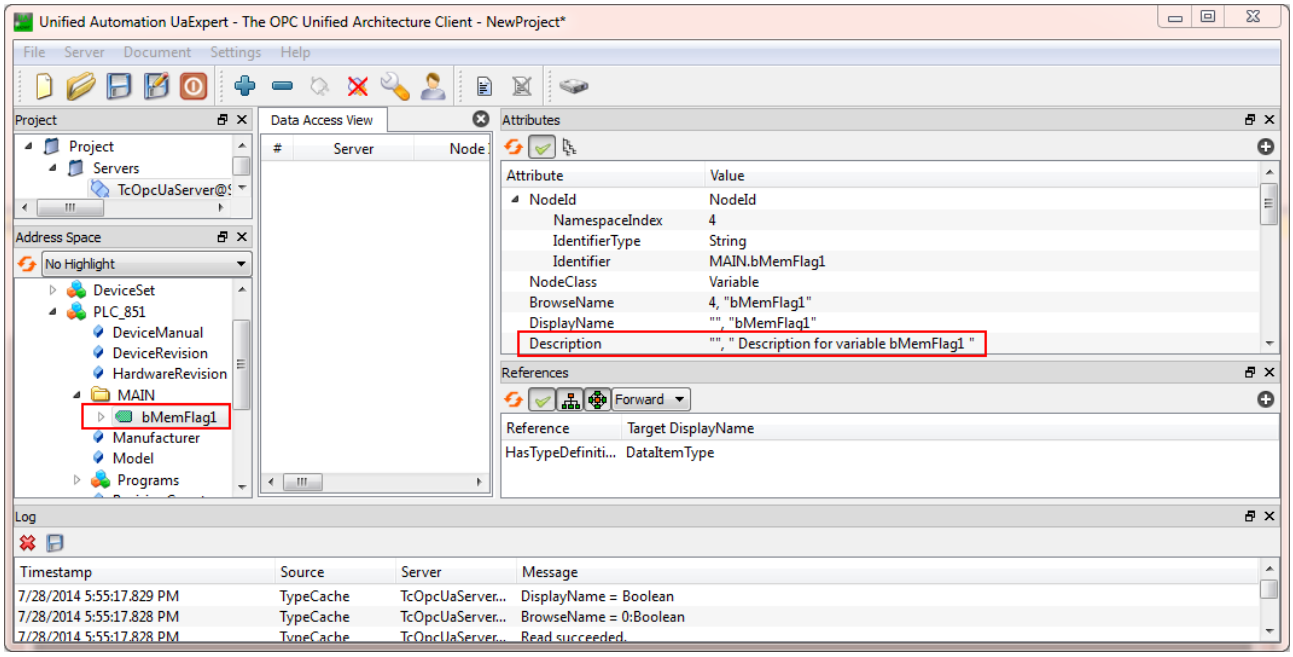
Um die Beschreibung einer Variablen im UA-Namensraum festzulegen, können Sie einen einfachen SPS-Kommentar hinter die entsprechende Variable einfügen, z.B.:

TwinCAT 3 (mit TMC Import):

```
{attribute 'OPC.UA.DA' := '1'}
bMemFlag1 : BOOL; (* Description for variable bMemFlag1 *)
```

TwinCAT 2 (mit TPY Import): Funktionalität nicht verfügbar

Dieser Kommentar wird automatisch zum Beschreibungsattribut des Knotens im UA-Namensraum.



Beispiel 2:
In diesem Beispiel sollen die beiden Instanzen fbTest1 und fbTest2 des Funktionsbausteins FB_BLOCK1 über OPC-UA verfügbar sein. Wird eine ganze Instanz freigegeben, dann sind auch alle in ihr enthaltenen Symbole über OPC-UA verfügbar. Das SPS-Programm könnte folgendermaßen aussehen, z.B.:

TwinCAT 3 (mit TMC Import):

```
PROGRAM MAIN
VAR
{attribute 'OPC.UA.DA' := '1'}
fbTest1 : FB_BLOCK1;
fbTest2 : FB_BLOCK1;
END_VAR
```

```
FUNCTION_BLOCK FB_BLOCK1
VAR_INPUT
{attribute 'OPC.UA.DA' := '1'}
ni1 : INT;
ni2 : INT;
END_VAR
VAR_OUTPUT
{attribute 'OPC.UA.DA' := '1'}
no1 : INT;
no2 : INT;
END_VAR
VAR
{attribute 'OPC.UA.DA' := '1'}
nx1 : INT;
nx2 : INT;
END_VAR
```

TwinCAT 2 (mit TPY Import):

```
PROGRAM MAIN
VAR
fbTest1 : FB_BLOCK1; (*~ (OPC:1:some description) *)
fbTest2 : FB_BLOCK1;
END_VAR
```

```
FUNCTION_BLOCK FB_BLOCK1
VAR_INPUT
ni1 : INT; (*~ (OPC:1:some description) *)
ni2 : INT;
END_VAR
VAR_OUTPUT
no1 : INT; (*~ (OPC:1:some description) *)
no2 : INT;
END_VAR
VAR
nx1 : INT; (*~ (OPC:1:some description) *)
nx2 : INT;
END_VAR
```

Die Instanz “fbTest1” wird für OPC-UA freigegeben, in Folge dessen alle in ihr enthaltenen Symbole ebenfalls automatisch für OPC-UA freigegeben werden, sprich fbTest.ni1, fbTest.ni2, usw. Die Instanz “fbTest2” ist nicht für OPC-UA markiert, allerdings wurden die drei in ihr enthaltenen Variablen - ni1, no1 und nx1 – im Funktionsbaustein markiert. Diese sind demzufolge in *allen* Instanzen über OPC-UA verfügbar.

Schritt 2: Herunterladen von Symbolbeschreibung in SPS-Projekt aktivieren

Die sogenannte Symboldatei enthält Informationen über alle in einem SPS-Projekt verfügbaren SPS-Variablen. Der OPC-UA Server benötigt diese Informationen für den Aufbau seines Namensraums. Die aktuellen Symbolinformationen werden standardmäßig automatisch in eine Symboldatei heruntergeladen. Diese befindet sich im Projektordner des entsprechenden TwinCAT Projekts. Damit die Symboldatei in die Ziel-Laufzeit übertragen wird, müssen Sie die Option ‘Download Symboldatei’ im SPS Projekt aktivieren.

TwinCAT 3:

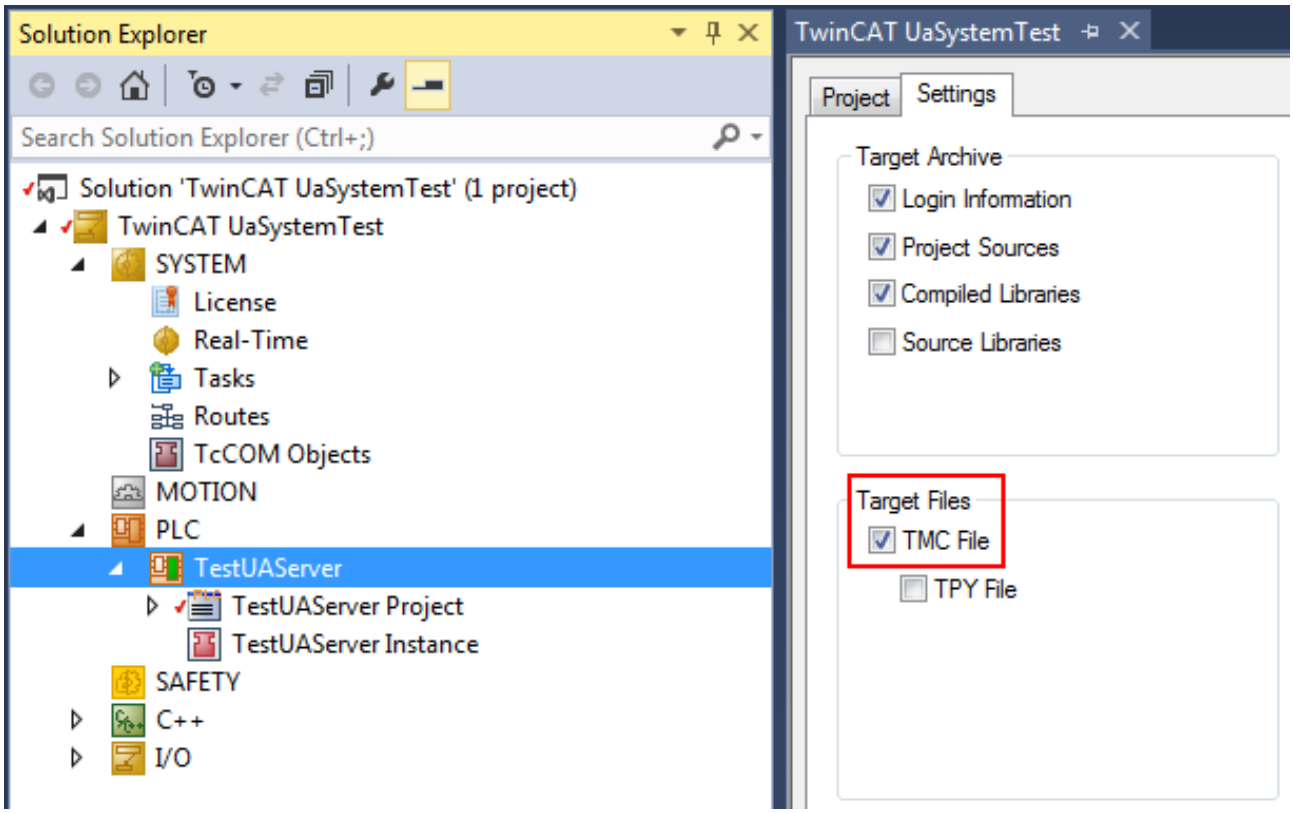
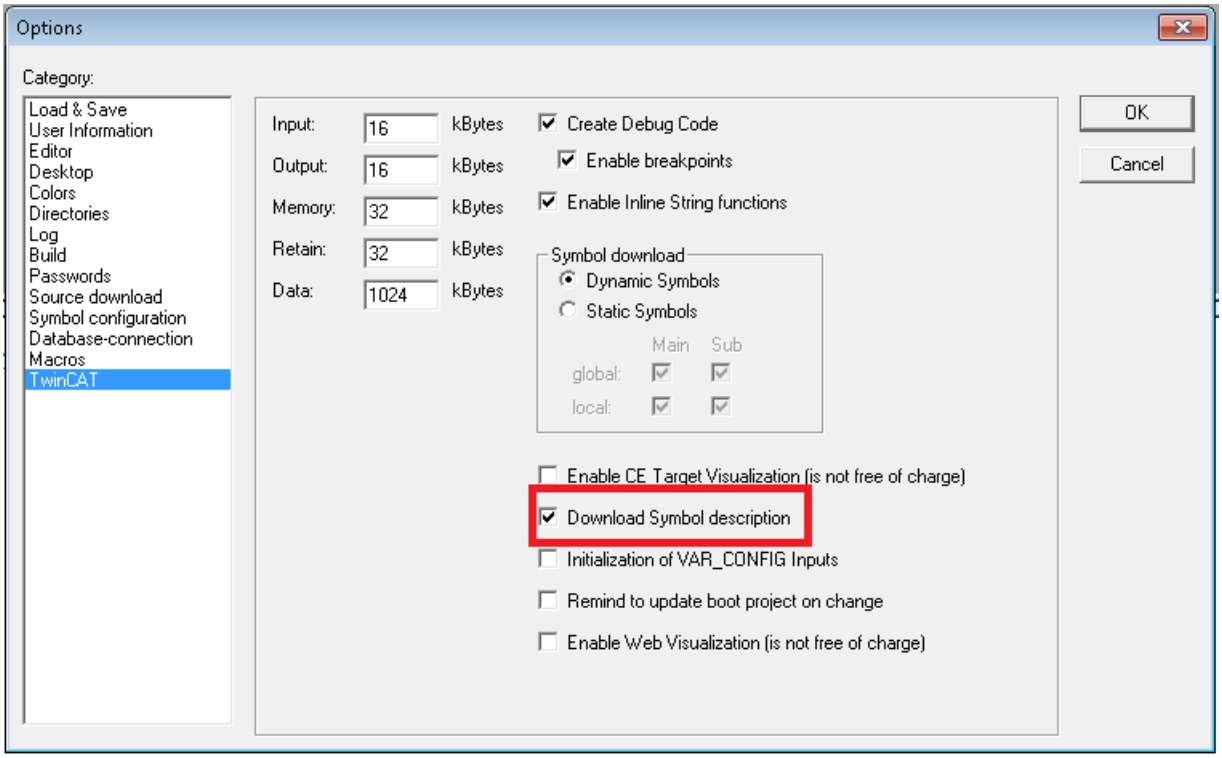


Abb. 1: **TwinCAT 2:**



Bei der Erstellung eines Bootprojekts wird die Symboldatei automatisch in das Bootverzeichnis der SPS-Laufzeit kopiert und dabei gibt der Dateiname Auskunft über den entsprechenden ADS-Port der SPS-Laufzeit.

Wenn sich OPC-UA Server und SPS-Laufzeit auf dem gleichen System befinden, müssen Sie nichts anderes tun, als den OPC-UA Server neu zu starten. Der OPC-UA Server wird automatisch so konfiguriert, dass er automatisch die Symboldatei aus dem TwinCAT Bootverzeichnis lädt und seinen Namensraum auf der Grundlage der in dieser Datei enthaltenen Symbolinformationen generiert. Wenn eine andere Symboldatei verwendet werden soll, dann müssen Sie entweder diese im OPC-UA Konfigurator oder direkt in der ServerConfig.xml Datei konfigurieren.

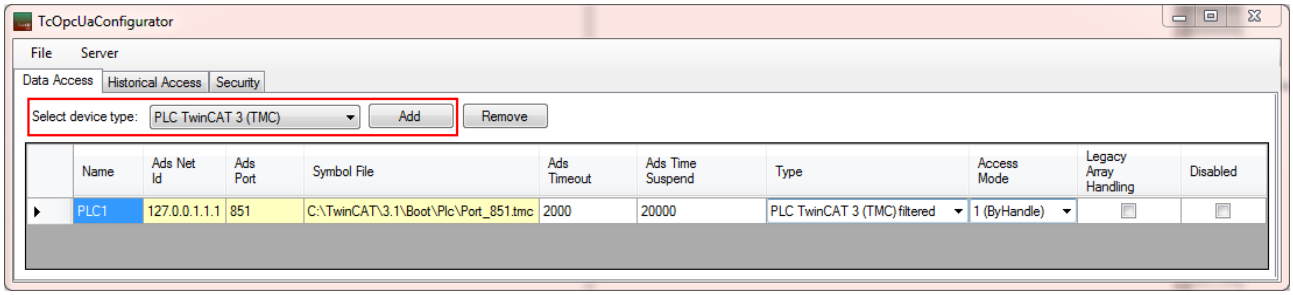
[optionaler] Schritt 3: Datenzugriffgerät in OPC-UA Server konfigurieren

Standardmäßig ist der OPC-UA Server so konfiguriert, dass er mit der ersten SPS-Laufzeit auf dem lokalen System in Verbindung tritt. Wenn Sie mehr als eine SPS-Laufzeit im UA-Namensraum veröffentlichen möchten, müssen Sie zusätzliche Datenzugriffsgeräte im OPC-UA Konfigurator konfigurieren. Diese Konfiguration ist möglicherweise bei folgenden Szenarien erforderlich:

- Sie haben eine andere (zweite oder dritte) SPS-Laufzeit, von der Sie Symbolinformationen an OPC-UA veröffentlichen möchten.
- Sie haben einen anderen Industrie-PC oder Embedded-PC, dessen SPS-Laufzeit und entsprechenden Symbolinformationen Sie OPC-UA öffentlich zugänglich machen möchten.

Lesen Sie bitte unseren Artikel Szenarien, worin verschiedene Anwendungsbeispiele die Möglichkeiten, wie der OPC-UA Server implementiert und verwendet werden kann, erläutert werden.

Um zusätzliche Geräte für den Datenzugriff zu konfigurieren, öffnen Sie bitte den OPC-UA Konfigurator und fügen neue Datenzugriffsgeräte hinzu.



Fahren Sie mit der Konfiguration aller notwendigen Parameter fort, wie oben unter Allgemeines beschrieben. Wenn Sie einen fernen Industrie-PC oder Embedded-PC konfigurieren, vergewissern Sie sich, dass Sie die richtigen Parameter für AdsNetId, AdsPort eingeben und die entsprechende Symboldatei des auf diesem PC laufenden SPS-Programms bereitstellen.

4.1.3.2 Historical Access

Dieser Artikel beschreibt die notwendigen Schritte der Konfiguration der Variablen im Namensraum des TwinCAT OPC-UA Servers für Historical Access (HA). Historical Access ist der Name einer Function von OPC-UA, bei der die Variablenwerte entweder dauerhaft auf einem Datenträger (Datei oder Datenbank) oder in der Geräte-RAM gespeichert werden, damit sie später vom Client abgerufen werden können. Die Art und Weise wie der OPC-UA Server die Variablenwerte liest und speichert kann konfiguriert werden.

Voraussetzungen

Es gelten die folgenden Voraussetzungen zur Nutzung von Historical Access:

- Das Laufzeitsystem, dessen Symbole für Historical Access gespeichert werden sollen, muss als Data Access konfiguriert sein. (vergleiche Dokumentationsartikel zum Data Access)
- Zur Verwendung einer SQL Compact Datenbank als Speichermedium muss die SQL Compact Runtime 3.5 SP2 auf dem OPC-UA Server installiert sein.
- Zur Verwendung einer SQL Server Datenbank muss mindestens OPC-UA Version 2.0 installiert sein.

Allgemeines

Die folgenden Schritte müssen einmal ausgeführt werden, um einen Werteverlauf von SPS-Variablen über OPC-UA Historical Access zu aktivieren:

- Aktivierung von Historical Access auf dem OPC-UA Server
- Aktivierung von Historical Access im TwinCAT SPS-Programm
- Aktivierung von Historical Access im TwinCAT C++ Modul
- Anzeige von Historical Access Werten in einem OPC-UA Client

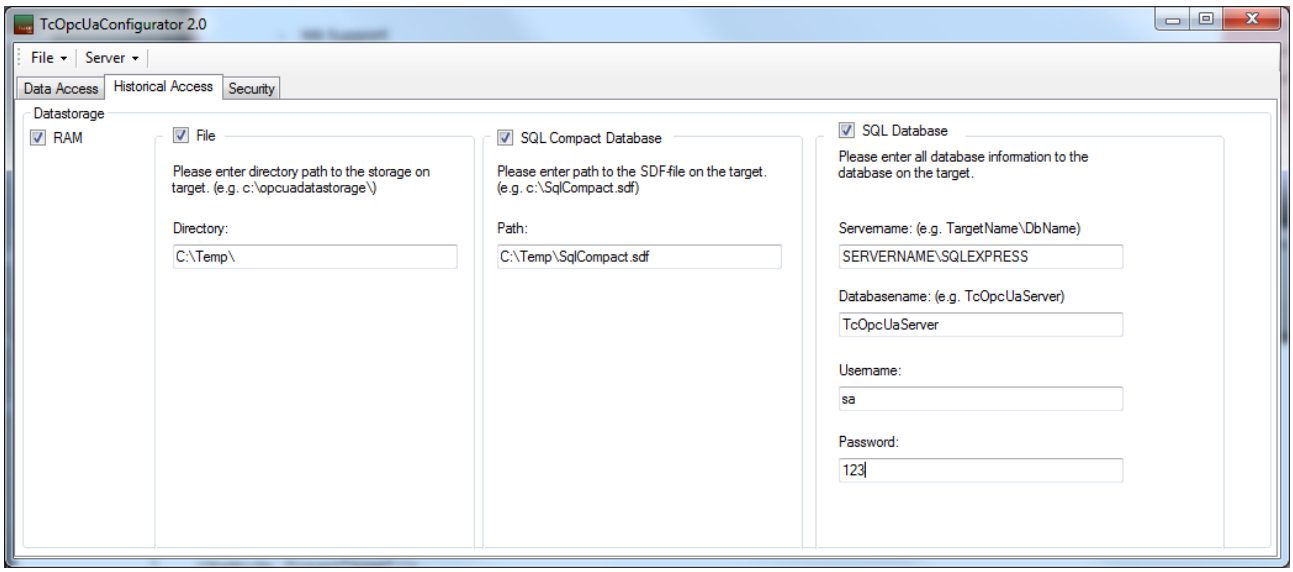
Diese Schritte werden nachfolgend ausführlicher beschrieben. Am Ende finden Sie weitere Auskünfte darüber, wie Sie über Historical Access gespeicherte Variablenwerte mit Hilfe der UA-Expert Software abrufen können.

Aktivierung von Historical Access auf dem UA Server

Es können verschiedene Speichermedien für die Speicherung von Werteverläufen verwendet werden. Diese Medien müssen einmal im OPC-UA Server aktiviert werden. Diese Einstellungen können entweder in der ServerConfig.xml Datei oder bequem mit Hilfe des OPC-UA Konfigurators vorgenommen werden. Das SPS-Programm bestimmt anschließend, welches Speichermedium zu verwenden ist. Nachdem ein oder mehrere Speichermedien auf dem OPC-UA Server aktiviert wurden, wählt das SPS-Programm das Speichermedium aus, dass für jedes Symbol zu verwenden ist (siehe Schritt 2 unten). Derzeit werden die folgenden vier Speichermedien unterstützt:

- Hauptspeicher: Speichert die Symbolwerte in der RAM des Geräts, auf dem der OPC-UA Server läuft.
- Datei: Speichert die Werte in einer Datei mit angegebenem Pfad.
- SQL Compact Database: Speichert die Werte in einer SQL Compact Database mit angegebenem Pfad. In diesem Fall muss die SQL Compact Runtime 3.5 SP2 als entsprechende Datenbankversion auf dem OPC-UA Server installiert sein.
- SQL Server Database: Speichert die Werte in einer SQL Server Database, die mit den Parametern Servername, Databasename, Username und Password spezifiziert wird.

Die folgenden Screenshots zeigen die Konfiguration dieser Modi mit Hilfe des OPC-UA Konfigurators und direkt über die ServerConfig.xml Datei.



Die unten gezeigten XML Tags müssen zwischen <OpcServerConfig><UaServerConfig> und </UaServerConfig></OpcServerConfig> in der ServerConfig.xml Datei eingefügt werden.

```
<HaModes>
<HaMode Type="RAM"/>
<HaMode Type="File" DataStore="C:\Temp\"/>
<HaMode Type="SQLServerCompact" DbName="C:\Temp\TcOpcUaServer.sdf"/>
<HaMode Type="SQLServer" Servername="SERVERNAME\SQLEXPRESS" DbName="TcOpcUaServer" User="sa"
Pass="123"/>
</HaModes>
```

Die für den jeweiligen Modus notwendigen Parameter sind:

Modus	Erforderliche Parameter
Speicher	Speichert die Werte in der RAM des Geräts, auf dem der OPC-UA Server läuft. Es sind keine weiteren Parameter erforderlich. Achtung: Nach einem Neustart des Geräts sind die gespeicherten Werte nicht mehr verfügbar. Das Speichermedium ist demzufolge kein dauerhaftes.
Datei	Speichert die Werte in mehreren Dateien, deren Speicherort mit dem Parameter DataStore festgelegt wird. Jedem für dieses Speichermedium konfigurierten Symbol (siehe Schritt 2 unten) wird dessen eigene Datei in diesem Verzeichnis zugeordnet. Darüber hinaus besteht eine Sicherungskopie der Datei für jedes Symbol, die dann erzeugt wird, sobald die Pufferlänge (siehe Schritt 2 unten) erreicht ist. Der Inhalt der Datendatei wird dann als Sicherungskopie gespeichert und es wird eine neue Datei erzeugt.
SQL Server Compact	Speichert die Werte in eine SQL Compact Database, dessen Speicherort mit dem Parameter DbName festgelegt wird. Es gelten die oben aufgeführten Anforderungen.
SQL Server	Speichert die Werte in einer SQL Server Database, die mit den Parametern Servername, DbName, User und Pass spezifiziert wird.

Aktivierung von Historical Access im TwinCAT SPS-Programm

Wenn eine SPS-Laufzeit als ein Datenzugriffsgerät konfiguriert ist (siehe entsprechenden Artikel in Dokumentation), dann werden die SPS-Variablen, die hierfür im SPS-Programm mit Hilfe von Kommentaren aktiviert wurden, automatisch durch den OPC-UA Server via OPC-UA verfügbar gemacht. Auf die gleiche Weise, wie Variablen für den Zugriff via OPC-UA freigegeben werden können, können einzelne Variablen für Historical Access freigegeben oder gesperrt werden. Dies geschieht auf ähnliche Weise im SPS-Programm mittels Eingabe von Kommentaren auf der rechten Seite eines Symbols. Die folgenden Parameter können hier spezifiziert werden:

TwinCAT 3 (mit TMC Import):

Parameter	Kommentar	Beschreibung
Aktivieren	{attribute 'OPC.UA.DA' := '1'}	Aktiviert das Symbol für Datenzugriff. Dies ist erforderlich, um das Symbol auch für Historical Access zur Verfügung zu stellen.
Für HA aktivieren	{attribute 'OPC.UA.HA' := '1'}	Aktiviert das Symbol für Historical Access.
Speichermedium	{attribute 'OPC.UA.HA.Storage' := '1'}	Legt das für die Speicherung der Symbolwerte zu verwendende Speichermedium fest (1 = RAM, 2 = Datei, 3 = SQL Compact Database, 4 = SQL Server Database).
Aufzeichnungsrate	{attribute 'OPC.UA.HA.Sampling' := '50'}	Definiert die Rate, mit der die Variablenwerte aus der Steuerung zu lesen und auf dem entsprechenden Speichermedium zu speichern sind. Der OPC-UA Server verwendet vordefinierte feste Aufzeichnungsraten, die hier verwendet werden können. Die Aufzeichnungsraten können gegebenenfalls in ServerConfig erweitert werden. Allerdings ist dies normalerweise nicht erforderlich, und es wird auch davon abgeraten, diese Einstellungen vorzunehmen.
Pufferlänge	{attribute 'OPC.UA.HA.Buffer' := '10000'}	Legt die Anzahl Elemente fest, die der Puffer aufnehmen soll.

TwinCAT 2 (mit TPY Import):

Parameter	Kommentar	Beschreibung
Aktivieren	(*~ (OPC:1:some description) *)	Aktiviert das Symbol für Datenzugriff. Dies ist erforderlich, um das Symbol auch für Historical Access zur Verfügung zu stellen.
Speichermedium	(*~ (OPC_UA_PROP[5000]:1:RAM) *)	Legt das für die Speicherung der Symbolwerte zu verwendende Speichermedium fest (1 = RAM, 2 = Datei, 3 = SQL Compact Database, 4 = SQL Server Database).
Aufzeichnungsrate	(*~ (OPC_UA_PROP[5000] [1]:50:SamplingRate) *)	Definiert die Rate, mit der die Variablenwerte aus der Steuerung zu lesen und auf dem entsprechenden Speichermedium zu speichern sind. Der OPC-UA Server verwendet vordefinierte feste Aufzeichnungsraten, die hier verwendet werden können. Die Aufzeichnungsraten können gegebenenfalls in ServerConfig erweitert werden. Allerdings ist dies normalerweise nicht erforderlich, und es wird auch davon abgeraten, diese Einstellungen vorzunehmen.
Pufferlänge	(*~ (OPC_UA_PROP[5000][2]:5000:Buffer) *)	Legt die Anzahl Elemente fest, die der Puffer aufnehmen soll.

Der folgende Auszug eines SPS-Programms beschreibt die Verwendung der oben aufgeführten Parameter unter Verwendung von vier SPS-Variablen:

TwinCAT 3 (mit TMC Import):

```
{attribute 'OPC.UA.DA' := '1'}
{attribute 'OPC.UA.HA' := '1'}
{attribute 'OPC.UA.HA.Storage' := '1'}
{attribute 'OPC.UA.HA.Sampling' := '5'}
{attribute 'OPC.UA.HA.Buffer' := '1000'}
_HistoryMem : UINT;

{attribute 'OPC.UA.DA' := '1'}
{attribute 'OPC.UA.HA' := '1'}
{attribute 'OPC.UA.HA.Storage' := '2'}
{attribute 'OPC.UA.HA.Sampling' := '20'}
{attribute 'OPC.UA.HA.Buffer' := '200'}
_HistoryFile : UINT;

{attribute 'OPC.UA.DA' := '1'}
{attribute 'OPC.UA.HA' := '1'}
{attribute 'OPC.UA.HA.Storage' := '3'}
{attribute 'OPC.UA.HA.Sampling' := '50'}
{attribute 'OPC.UA.HA.Buffer' := '10000'}
_HistoryDBcompact : UINT;

{attribute 'OPC.UA.DA' := '1'}
{attribute 'OPC.UA.HA' := '1'}
{attribute 'OPC.UA.HA.Storage' := '4'}
{attribute 'OPC.UA.HA.Sampling' := '5'}
{attribute 'OPC.UA.HA.Buffer' := '1000000'}
_HistoryDB : UINT;
```

TwinCAT 2 (mit TPY Import):

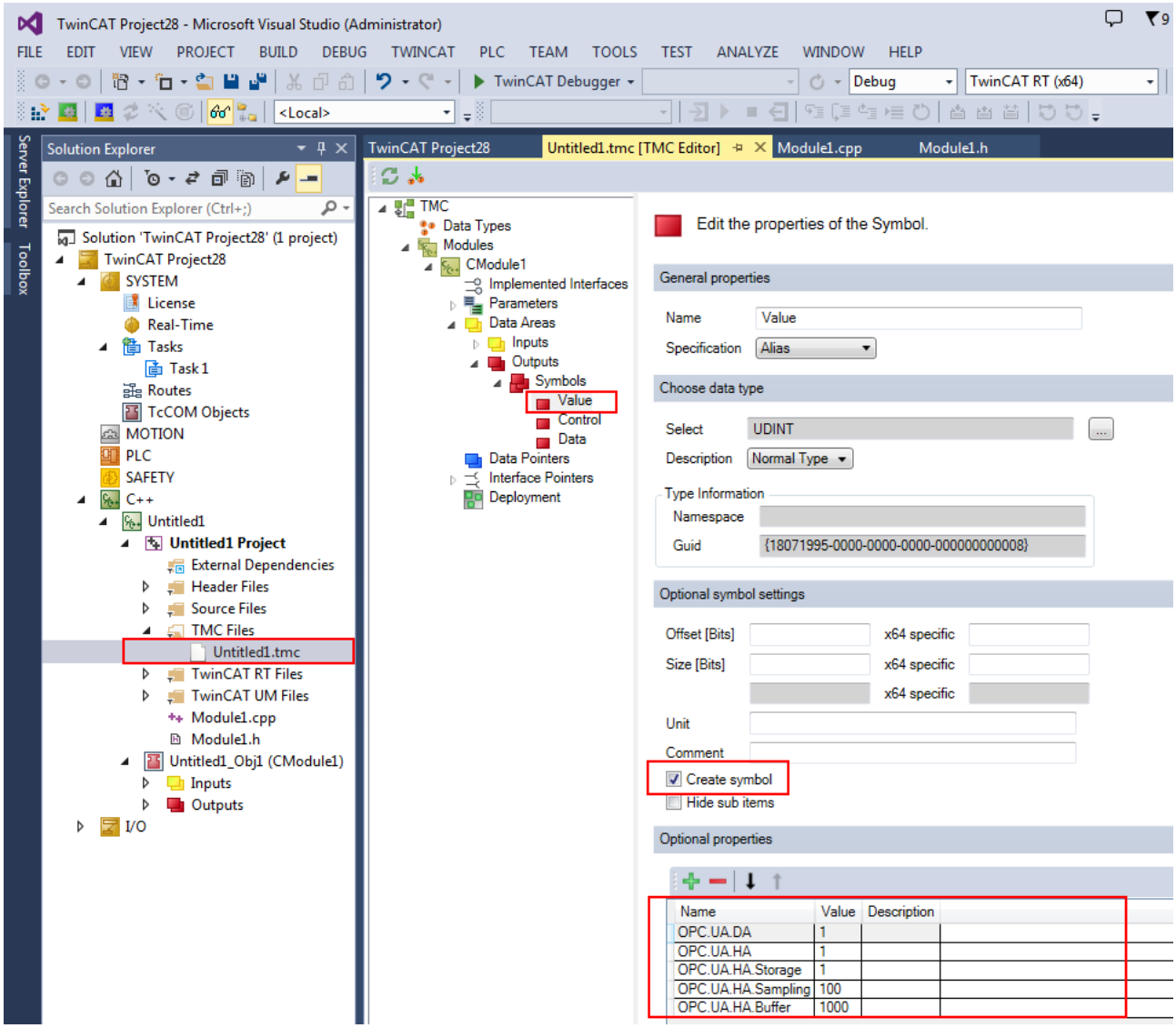
```
_HistoryMem : UINT; (*~ (OPC:1:available for OPC-UA Clients)
(OPC_UA_PROP[5000]:1:Memory)
(OPC_UA_PROP[5000][1]:5:SamplingRate)
(OPC_UA_PROP[5000][2]:1000:Buffer) *)

_HistoryFile : UINT; (*~ (OPC:1:available for OPC-UA Clients)
(OPC_UA_PROP[5000]:2:File)
(OPC_UA_PROP[5000][1]:20:SamplingRate)
(OPC_UA_PROP[5000][2]:200:Buffer) *)

_HistoryDBcompact : UINT; (*~ (OPC:1:available for OPC-UA Clients)
(OPC_UA_PROP[5000]:3:SQLCompact)
(OPC_UA_PROP[5000][1]:50:SamplingRate)
(OPC_UA_PROP[5000][2]:10000:Buffer) *)

_HistoryDB : UINT; (*~ (OPC:1:available for OPC-UA Clients)
(OPC_UA_PROP[5000]:4:SQL)
(OPC_UA_PROP[5000][1]:5:SamplingRate)
(OPC_UA_PROP[5000][2]:1000000:Buffer) *)
```

Aktivierung von Historical Access im TwinCAT C++ Modul

Jedes TwinCAT 3 C++ Symbol, dessen Verlauf aufgezeichnet werden soll, muss entsprechend konfiguriert werden. Vergleichbar mit der Konfiguration von SPS-Symbolen für Historical Access (siehe oben), müssen die gleichen Parameter für ein C++ Symbol konfiguriert werden, wie die folgende Tabelle zeigt:

Parameter	Kommentar	Beschreibung
Aktivieren	OPC.UA.DA := 1	Aktiviert das Symbol für Datenzugriff. Dies ist erforderlich, um das Symbol auch für Historical Access zur Verfügung zu stellen.
HA aktivieren	OPC.UA.HA := 1	Aktiviert das Symbol für Historical Access.
Speichermedium	OPC.UA.HA.Storage := 1	Legt das für die Speicherung der Symbolwerte zu verwendende Speichermedium fest (1 = RAM, 2 = Datei, 3 = SQL Compact Database, 4 = SQL Server Database).
Aufzeichnungsrate	OPC.UA.HA.Sampling := 50	Definiert die Rate, mit der die Variablenwerte aus der Steuerung zu lesen und auf dem entsprechenden Speichermedium zu speichern sind. Der OPC-UA Server verwendet vordefinierte feste Aufzeichnungsraten, die hier verwendet werden können. Die Aufzeichnungsraten können gegebenenfalls in ServerConfig erweitert werden. Allerdings ist dies normalerweise nicht erforderlich, und es wird auch davon abgeraten, diese Einstellungen vorzunehmen.
Pufferlänge	OPC.UA.HA.Buffer := 10000	Legt die Anzahl Elemente fest, die der Puffer aufnehmen soll.

Allerdings müssen in TwinCAT 3 C++ diese Parameter im TMC Code Editor des entsprechenden C++ Moduls konfiguriert werden. Navigieren Sie mit Hilfe des TMC Code Editors zu dem Symbol, das für Historical Access freigegeben werden soll, und fügen diese Parameter in den Bereich “Optionale Eigenschaften” der Symboleigenschaften hinzu. Vergewissern Sie sich, dass Sie das Kontrollkästchen „Symbol erzeugen“ aktiviert haben.



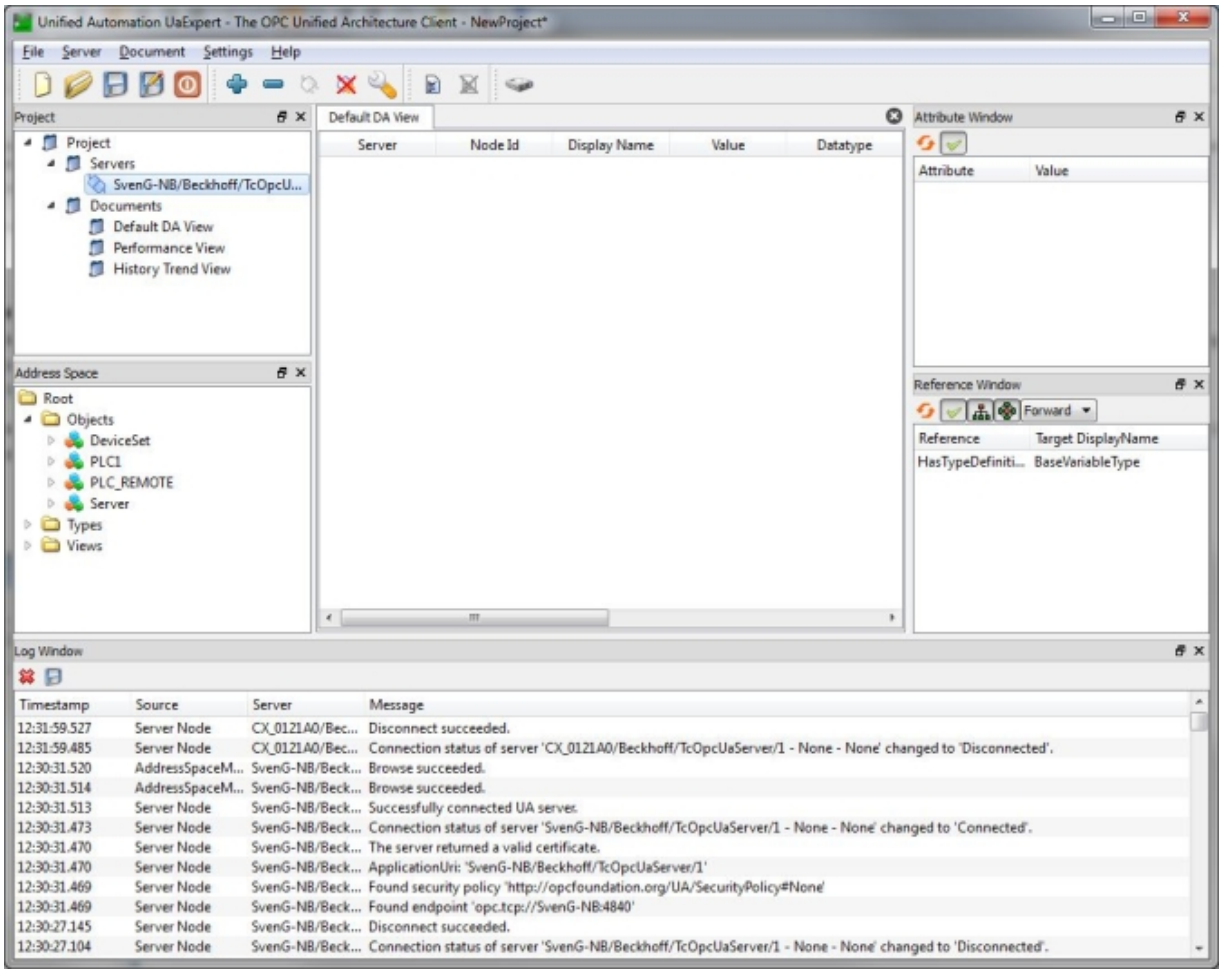
Anzeige von Historical Access Werten in einem OPC-UA Client

Das folgende Beispiel basiert auf obigem SPS-Programm, das ebenfalls als Download im Abschnitt Beispiele dieser Dokumentation zur Verfügung steht. Darüber hinaus ist eine für Historical Access konfigurierte ServerConfig.xml Datei erforderlich, siehe Schritt 1 oben. Führen Sie bitte folgende Schritte aus:

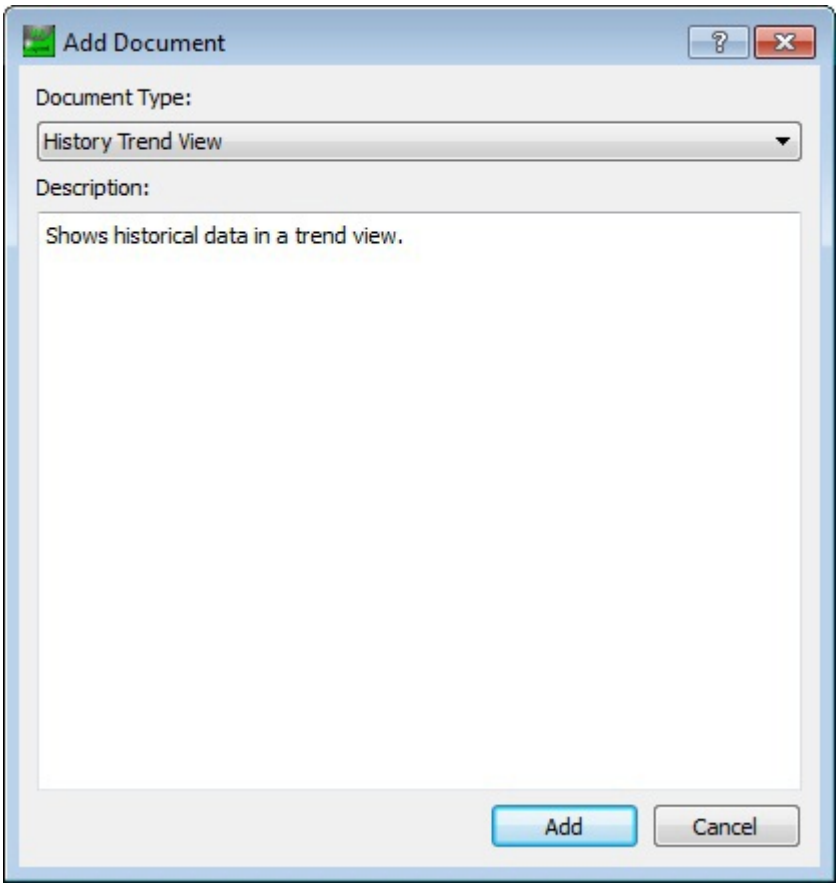
1. Öffnen Sie das SPS-Programm in TwinCAT XAE indem Sie eine neue TwinCAT-Lösung erstellen und importieren dann die tpzip Datei als SPS-Projekt in diese.
2. Aktivieren Sie den TMC/TPY Download in den SPS-Projekteigenschaften
3. Die Konfiguration aktivieren

   ⇨ Der OPC-UA Server ruft nun Symbolwerte mit den spezifizierten Parametern (Speichermedium, Aufzeichnungsrate, Pufferlänge) ab. Sie können anschließend diese Daten mit einem OPC-UA Client anzeigen, der Historical Access unterstützt, z.B. mit Hilfe der UA-Expert Software von Unified Automation.
   Die folgenden Tests werden mit UA-Expert Version 1.2.0.132 durchgeführt.
4. Vergewissern Sie sich, dass Ihr System den Anforderungen genügt (siehe Anfang von Artikel).

5. “UA-Expert” Software starten und Verbindung mit dem OPC-UA Server herstellen

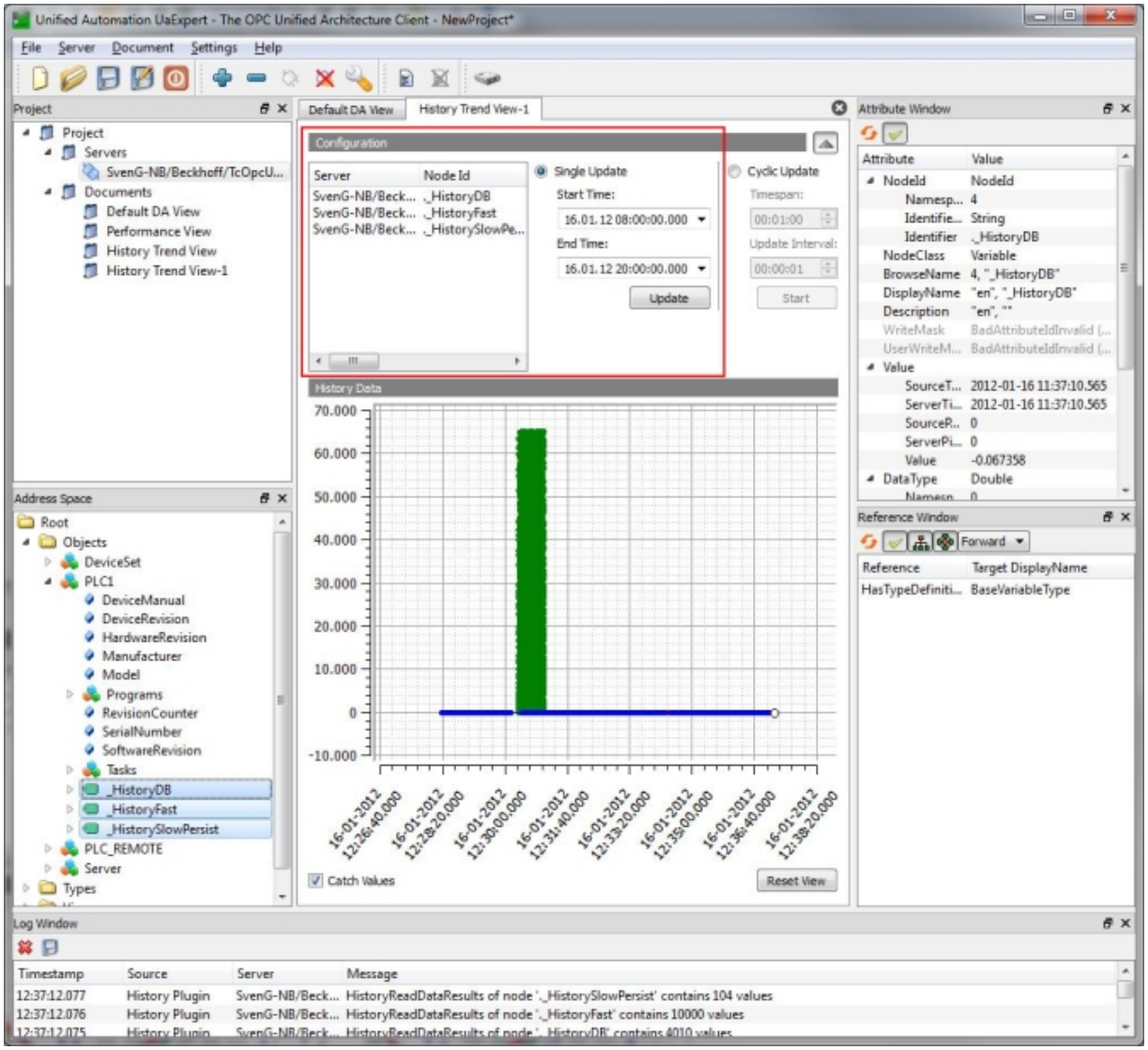

Unified Automation UaExpert - The OPC Unified Architecture Client - NewProject*
File Server Document Settings Help
Project
Project
Servers
SvenG-NB/Beckhoff/TcOpcU...
Documents
Default DA View
Performance View
History Trend View
Default DA View
Server Node Id Display Name Value Datatype
Attribute Window
Attribute Value
Address Space
Root
Objects
DeviceSet
PLC1
PLC_REMOTE
Server
Types
Views
Reference Window
Forward
Reference Target DisplayName
HasTypeDefiniti... BaseVariableType
Log Window
Timestamp Source Server Message
12:31:59.527 Server Node CX_0121A0/Bec... Disconnect succeeded.
12:31:59.485 Server Node CX_0121A0/Bec... Connection status of server 'CX_0121A0/Beckhoff/TcOpcUaServer/1 - None - None' changed to 'Disconnected'.
12:30:31.520 AddressSpaceM... SvenG-NB/Beck... Browse succeeded.
12:30:31.514 AddressSpaceM... SvenG-NB/Beck... Browse succeeded.
12:30:31.513 Server Node SvenG-NB/Beck... Successfully connected UA server.
12:30:31.473 Server Node SvenG-NB/Beck... Connection status of server 'SvenG-NB/Beckhoff/TcOpcUaServer/1 - None - None' changed to 'Connected'.
12:30:31.470 Server Node SvenG-NB/Beck... The server returned a valid certificate.
12:30:31.470 Server Node SvenG-NB/Beck... ApplicationUri: 'SvenG-NB/Beckhoff/TcOpcUaServer/1'
12:30:31.469 Server Node SvenG-NB/Beck... Found security policy 'http://opcfoundation.org/UA/SecurityPolicy#None'
12:30:31.469 Server Node SvenG-NB/Beck... Found endpoint 'opc.tcp://SvenG-NB:4840'
12:30:27.145 Server Node SvenG-NB/Beck... Disconnect succeeded.
12:30:27.104 Server Node SvenG-NB/Beck... Connection status of server 'SvenG-NB/Beckhoff/TcOpcUaServer/1 - None - None' changed to 'Disconnected'.

6. Neue “History Trend Ansicht” hinzufügen



7. Den PLC1-Namensraum durchsuchen und die SPS-Variablen _HistoryDB, _HistoryDBcompact, _HistoryFast und _HistorySlowPersist mittels Drag & Drop hinzufügen.

8. Sie können nun den gewünschten Zeitraum, für den die Symbolwerte angezeigt werden sollen, mit Hilfe der Steuerelemente StartTime und EndTime festlegen, oder falls erforderlich ein CyclicUpdate für diese Variablen starten.

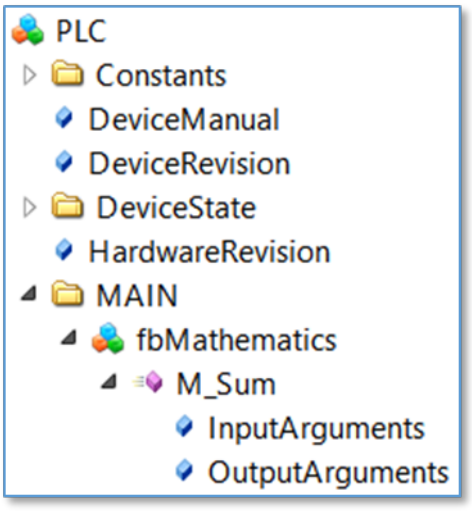


Hier sollte noch mal daran erinnert werden, dass für die Verwendung eines Datenspeichermediums bestimmte Anforderungen gelten, die am Anfang des Artikels beschrieben wurden.

4.1.3.3 Methodenaufruf

Methodenaufrufe sind ein sehr grundlegender Teil der OPC-UA Spezifikation. Mit der Einführung dieser Funktionalitäten in die SPS-Welt bietet TwinCAT 3 den Kunden eine sehr effiziente Weise zur Ausführung von RPC-Aufrufen in die IEC61131-Welt und verringert somit die klassischen Handshake-Pattern bei der Kommunikation zwischen den Geräten. Der TwinCAT OPC-UA Server importiert SPS-Methoden wie OPC-UA Methoden über seinen TMC-Import.

Beachten Sie dabei, dass, wenngleich die SPS-Methode eine normale Methode im UA-Namensraum zu sein scheint, sie immer noch eine IEC61131-Methode ist, die innerhalb des Echtzeit-Kontexts ausgeführt wird - und somit unter den Kontext einer Echtzeit-Task fällt! Der SPS-Entwickler muss daher Vorsichtsmaßnahmen treffen, dass die Ausführungszeit der Methode in die Task-Zykluszeit passt!



Methoden in IEC61131-3

Methoden in der IEC61131-Welt werden immer unterhalb eines Funktionsblocks konfiguriert. Auf Hochsprachenebene kann der Funktionsblock als umgebende Klasse der Methode betrachtet werden. Die Methode selbst muss mit einem speziellen SPS-Attribut deklariert werden, sodass das TwinCAT System weiß, dass die Methode für einen Remote-Methodenaufruf aktiviert werden soll.

```
{attribute 'TcRpcEnable':='1'}
METHOD M_Sum : INT
VAR_INPUT
  a : INT;
  b : INT;
END_VAR
```

Gefilterter oder ungefilterter Modus

In Abhängigkeit dessen, welcher Importmodus verwendet wird, muss der umgebende Funktionsblock ebenfalls für den OPC-UA Zugriff aktiviert werden. Dies kann durch Verwendung der normalen SPS-Attribute für den OPC-UA Zugriff erfolgen, wie dies in anderen Teilen dieser Dokumentation beschrieben ist.

Beispiel: Die Methode M_Sum befindet sich im Funktionsblock FB_Mathematics. Die Deklaration der Funktionsblockinstanz verwendet das SPS-Attribut, das den Funktionsblock - und damit die Methode ebenfalls - für den OPC-UA Zugriff aktiviert hat.

```
PROGRAM MAIN
VAR
  {attribute 'OPC.UA.DA':='1'}
  fbMathematics : FB_Mathematics;
END_VAR
```

Beachten Sie dabei, dass Sie nur dann die SPS-Attribute verwenden müssen, wenn der gefilterte Modus verwendet wird, um Symbole aus der SPS für den Zugriff über OPC-UA freizugeben. Dies ist die Standardeinstellung des OPC-UA Servers.

4.1.3.4 Typsystem

Einer der größten Vorteile von OPC-UA ist das Meta-Modell, welches zur Bereitstellung von Basistypen genauso verwendet werden kann wie zur Erweiterung des Typsystems durch eigene Modelle. Der selbe Mechanismus wird zur Darstellung realer Objekte (Nodes) verwendet, so dass OPC-UA Clients einen Objekttyp bestimmen können.

Der TwinCAT 3 OPC-UA Server veröffentlicht Typinformationen von der IEC61131-Welt in seinen Namensraum. Dies umfasst nicht nur Basistypen wie z. B. BOOL, INT, DINT oder REAL, sondern auch erweiterte Typinformationen, wie z. B. die aktuelle Klasse (Funktionsblock) oder Struktur, die ein Objekt repräsentiert.

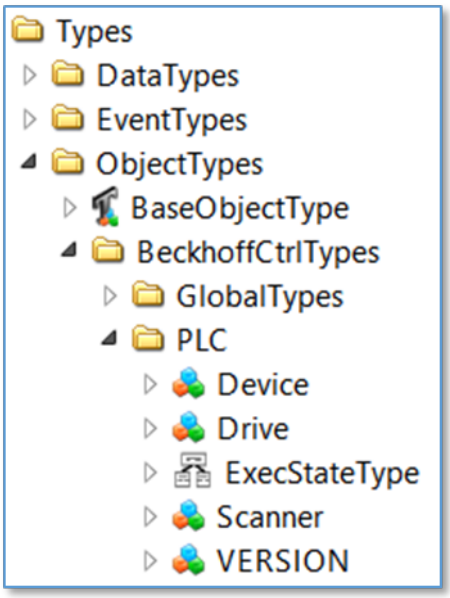
Standort der Typinformationen

Typinformationen sind Bestandteil des UA-Namensraums. Der TwinCAT OPC-UA Server erweitert die Basistypinformationen wie folgt:

Lokale Typinformationen (die nur für eine Laufzeit gültig ist) werden in demselben Namensraum wie die Laufzeitsymbole gespeichert.

Globale Typinformationen (die über verschiedene Laufzeiten gültig sein können), werden in einem eigenen globalen Namensraum gespeichert.

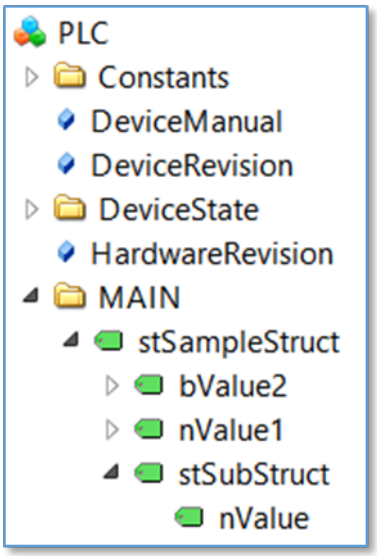
Das Typsystem ist ebenfalls virtuell verfügbar und kann in Types Sektion des OPC-UA Servers eingesehen werden:



Jeder Nichtstandard-Datentyp wird im Verzeichnis BeckhoffCtrlTypes eingetragen.

Grundlagen

Angenommen, die TwinCAT 3 SPS besteht aus einem SPS-Programm mit verschiedenen STRUCTs. Jeder STRUCT wird als Knoten in einem UA Namensraum repräsentiert, mit jedem Element der Struktur als untergeordneten Knoten.

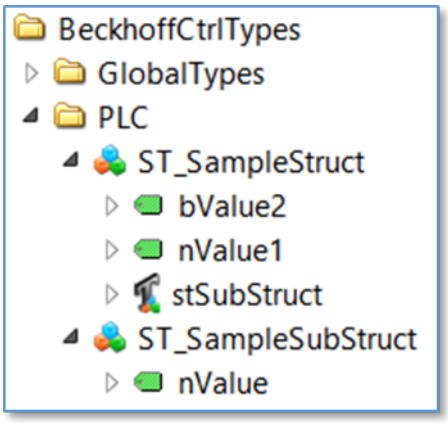


In dem vorstehenden Beispiel besteht der STRUCT stSampleStruct aus drei untergeordneten Elementen: einer Variable nValue1 des Typs INT, einer Variable bValue2 des Typs BOOL und einem weiteren STRUCT stSubStruct, der lediglich ein untergeordnetes Element enthält – eine Variable nValue des Typs INT.

Die Struktur selbst ist lediglich eine reguläre Variable im UA Namensraum, die den Datentyp ByteString hat. Die Clients können daher einfach mit dem Wurzelelement (der Struktur selbst) verbunden werden und dessen Werte durch Interpretation des ByteString gelesen/geschrieben werden. Damit die Interpretation jedes untergeordneten Elements vereinfacht wird, enthält das Typsystem mehr Informationen über die Struktur selbst, in erster Linie in der Referenz zur Instanz:

Reference	Target DisplayName
HasTypeDe...	ST_SampleStruct
HasCompo...	nValue1
HasCompo...	bValue2
HasCompo...	stSubStruct

Dann enthält das Typsystem mehr Informationen über ST_SampleStruct:



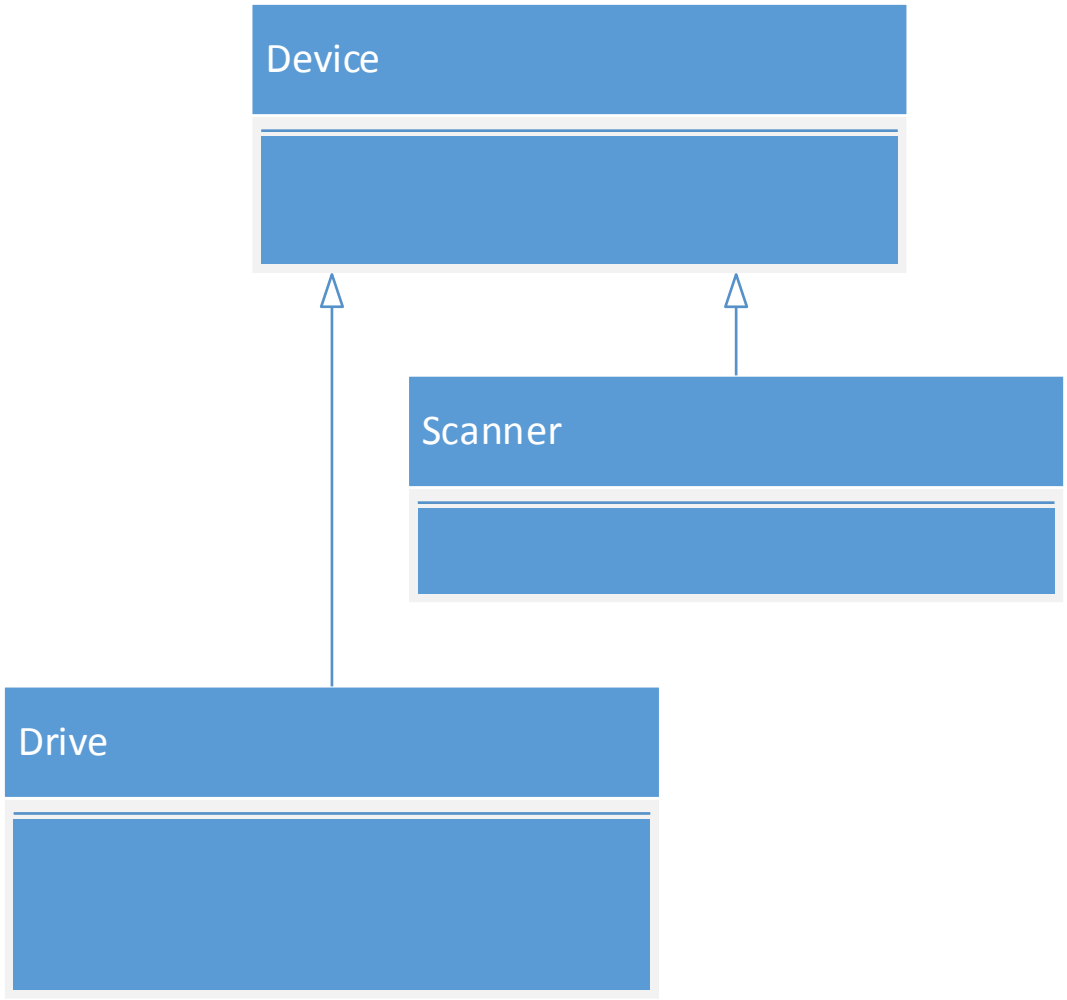
Und in den Referenzen von ST_SampleStruct:

Reference	Target DisplayName
HasTypeDe...	DataItemType
HasCompo...	nValue1
HasCompo...	bValue2
HasCompo...	stSubStruct

Tatsache ist, wenn ein Client die ByteString Darstellung eines Strukturwurzelknotens lesen möchte, kann das Typsystem sehr nützliche Informationen darüber bieten, wie jede Position des ByteStrings zu interpretieren ist. In dem vorstehenden Beispiel „sieht“ der Client tatsächlich, dass das erste Byte von ST_SampleStruct eine BOOL-Variable repräsentiert, das zweite und dritte Byte repräsentieren eine INT-Variable und das vierte und fünfte Byte eine Variable des Datentyps INT.

Objektorientierte Erweiterungen

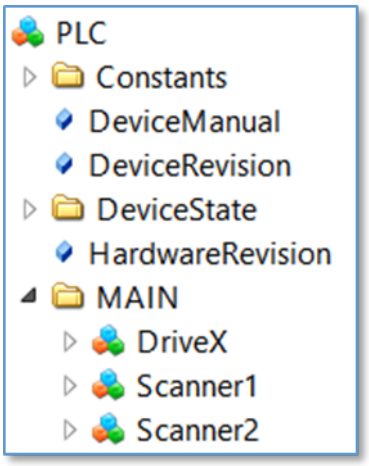
Angenommen, dass die TwinCAT 3 SPS-Laufzeit ein SPS-Programm enthält, dessen Struktur wie folgt visualisiert werden kann:



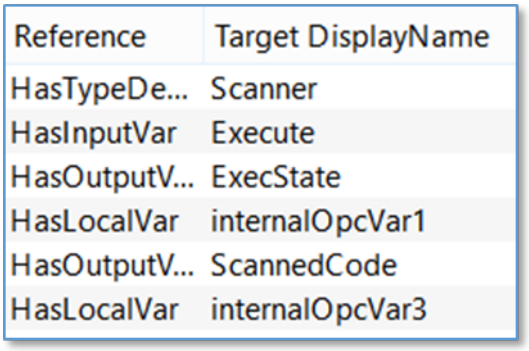
Die zwei Funktionsblöcke Scanner und Drive sind von der Basisklasse Device abgeleitet, indem objektorientierte Erweiterungen der IEC61131-3 verwendet werden. Das MAIN-Programm enthält nun die folgenden Deklarationen:

```
PROGRAM MAIN
VAR
  {attribute 'OPC.UA.DA':='1'}
  Scanner1 : Scanner;
  {attribute 'OPC.UA.DA':='1'}
  Scanner2 : Scanner;
  {attribute 'OPC.UA.DA':='1'}
  DriveX : Drive;
END_VAR
```

Alle drei Objekte sind vom Typ Device, aber auch von ihrem speziellen Datentyp. Der OPC-UA Server importiert nun die Objekte wie folgt:



Der zugrundeliegende Datentyp kann nun in der Referenz jedes Objekts bestimmt werden, z. B. Objekt Scanner1:

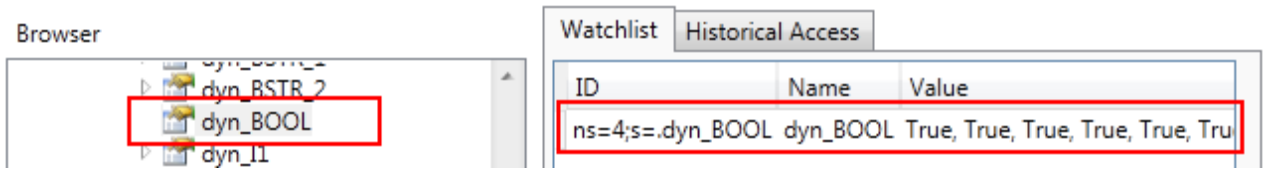


Reference	Target DisplayName
HasTypeDe...	Scanner
HasInputVar	Execute
HasOutputV...	ExecState
HasLocalVar	internalOpcVar1
HasOutputV...	ScannedCode
HasLocalVar	internalOpcVar3

Entsprechend des zugrundeliegenden IEC61131-Programms ist das Objekt Scanner1 vom Datentyp Scanner und zeigt ebenfalls, welche Variablen enthalten sind und welcher Art die Variabel ist: Eine Eingangs-, Ausgangs- oder interne (lokale) Variable. Ein schneller Blick auf das Diagramm oben zeigt, dass nicht nur die Variablen des tatsächlichen Funktionsblocks hier wiedergegeben werden, sondern auch die abgeleiteten Variablen der Basisklasse. Die gesamte IEC61131-3 Vererbungskette wird im UA Namensraum repräsentiert.

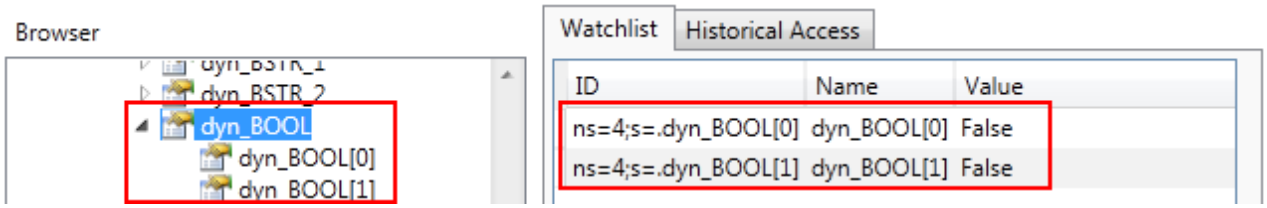
4.1.3.5 Verwendung von Arrays

Standardmäßig werden Arrays im UA-Namensraum als einzelner Knoten betrachtet. Dies bedeutet, dass, wenn Sie z.B. in der SPS ein Array *dyn_BOOL[10]* definieren (und dieses auch für OPC-UA freigegeben haben), es anschließend im UA-Namensraum wie folgt auftaucht:

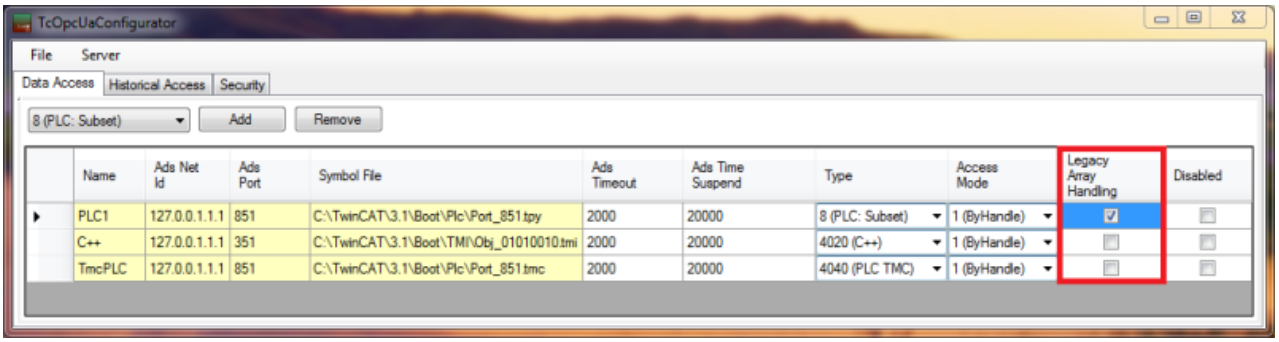


Der Vorteil dieser Vorgehensweise ist eine deutliche Reduzierung der Komplexität des UA-Namensraums UND des Speicherverbrauchs, da nicht jede Position eines Arrays als einzelner Knoten im Namensraum zur Verfügung gestellt werden muss. Moderne UA-Clients können jedoch weiterhin auf die einzelnen Array-Positionen über den sogenannten "RangeOffset" zugreifen.

Damit jedoch auch ältere UA-Clients unterstützt werden, welche dieses Feature nicht bieten, können Sie die Positionen eines Arrays auch als einzelne Knoten im UA-Namensraum verfügbar machen. Dies würde sich dann wie folgt darstellen:



Diese Einstellung ist durch Aktivierung der Option "**Legacy Array Handling**" im UA-Configurator innerhalb der jeweiligen Namensraum-Konfiguration (z.B. UaServer\UaServerConfig\Namespaces\PLC1) verfügbar.



Alternativ können Sie auch die Konfigurationsdatei [▶ 78] manuell editieren und dieser den folgenden Knoten hinzufügen:

```
<UaNodeManager>
  <Name>PLC1</Name>
  <AdsPort>851</AdsPort>
  <AdsNetId>127.0.0.1.1.1</AdsNetId>
  <AdsTimeout>2000</AdsTimeout>
  <AdsTimeSuspend>20000</AdsTimeSuspend>
  <AutoCfg>8</AutoCfg>
  <AutoCfgSymFile>C:\TwinCAT\Boot\CurrentConfig_1.tpy</AutoCfgSymFile>
  <ArraySubItemLegacySupport>1</ArraySubItemLegacySupport>
  <Disabled>0</Disabled>
</UaNodeManager>
```

Bitte beachten Sie jedoch hierbei, dass je nach Umfang des SPS-Projekts, der UA-Namensraum hierdurch deutlich an Komplexität gewinnt, was sich wiederum in einer erhöhten Speicherauslastung des UA-Servers widerspiegelt.

Eine Änderung der oben genannten Einstellungen wird erst aktiv, wenn Sie den UA-Server neu starten.

4.1.3.6 Liste der Attribute und Kommentare

Die Konfiguration der Laufzeitvariablen für verschiedene OPC-UA Funktionalitäten (z. B. Data Access oder Historical Access) erfolgt direkt vom SPS-Programm oder vom TMC Code Editor aus (bei Verwendung von TwinCAT 3 C++). Der Vorteil besteht darin, dass ein SPS-Entwickler direkt im Programmcode, mit dem er vertraut ist, entscheiden kann, ob und wie eine Variable für OPC-UA freizugeben ist. Eine derartige Aktivierung wird dann mittels Einfügen eines Kommentars zusammen mit dem entsprechenden OPC-UA-Tag vor der Variablen vorgenommen, z.B.:

TwinCAT 3 SPS (TMC):

```
{attribute 'OPC.UA.DA' := '1'}
bVariable : BOOL;
```

TwinCAT 2 SPS (TPY):

```
bVariable : BOOL; (*~ (OPC:1:available)*)
```

Eine detaillierte Beschreibung darüber, wie Sie Attribute und Kommentare verwenden können, erhalten Sie in dem entsprechenden Kapitel der Laufzeitkomponenten, die verwendet werden (PLC, C++, I/O Task, ...).

In der folgenden Tabelle wird ein Überblick über alle definierbaren Tags und deren Bedeutung gegeben. Siehe jeweiligen Unterabschnitt der entsprechenden Funktion für eine ausführliche Beschreibung des Funktionsprinzips.

Tab. 6: TwinCAT 3 (TMC):

OPC-UA Funktion	SPS Tag	C++ TMC Code Editor (Optionale Eigenschaften)	Bedeutung
Data Access (DA)	{attribute 'OPC.UA.DA' := '0'}	Name: OPC.UA.DA Wert: 0	Sperrt eine Variable für OPC-UA, woraufhin diese im UA-Namensraum nicht mehr sichtbar ist.
Data Access (DA)	{attribute 'OPC.UA.DA' := '1'}	Name: OPC.UA.DA Wert: 1	Aktiviert eine Variable für OPC-UA, woraufhin diese im UA-Namensraum sichtbar wird. Dieser Tag muss immer gesetzt werden, wenn eine Variable für UA verwendet werden soll.
Data Access (DA)	{attribute 'OPC.UA.DA.Access' := 'x'}	Name: OPC.UA.DA.Access Wert: siehe rechte Spalte	Bestimmt den Lese-/Schreib-Zugriff für eine Variable, in Abhängigkeit des Parameters „x“. 0 = Kein 1 = Schreibgeschützt 2 = Nur Schreibzugriff 3 = Lese- und Schreibzugriff (Standardwert, wenn Tag nicht verwendet wird)
Historical Access (HA) [▶ 45]	{attribute 'OPC.UA.HA' := '1'}	Name: OPC.UA.HA Wert: 1	Aktiviert eine Variable für “Historical Access”. Muss zusammen mit {attribute 'OPC.UA.DA' := '1'} verwendet werden.
Historical Access (HA) [▶ 45]	{attribute 'OPC.UA.HA.Storage' := 'x'}	Name: OPC.UA.HA.Storage Wert: siehe rechte Spalte	Definiert den Speicherort für Historical Access, in Abhängigkeit des Parameters “x” 1 = Speicher 2 = Datei 3 = SQL Compact Database 4 = SQL Server Database
Historical Access (HA) [▶ 45]	{attribute 'OPC.UA.HA.Sampling' := 'x'}	Name: OPC.UA.HA.Sampling Wert: siehe rechte Spalte	Legt die Abtastrate fest, mit der die Variablenwerte zu speichern sind, in Abhängigkeit des Parameters „x“ in [ms]
Historical Access (HA) [▶ 45]	{attribute 'OPC.UA.HA.Buffer' := 'x'}	Name: OPC.UA.HA.Buffer Wert: siehe rechte Spalte	Legt die maximale Anzahl im Datenspeicher zu haltender Werte, in Abhängigkeit des Parameters „x“.

Tab. 7: TwinCAT 2 (TPY):

OPC-UA Funktion	SPS Tag	Bedeutung
Data Access (DA)	(*~ (OPC:0:not available) *)	Sperrt eine Variable für OPC-UA, woraufhin diese im UA-Namensraum nicht mehr sichtbar ist.
Data Access (DA)	(*~ (OPC:1:available) *)	Aktiviert eine Variable für OPC-UA, woraufhin diese im UA-Namensraum sichtbar wird. Dieser Tag muss immer gesetzt werden, wenn eine Variable für UA verwendet werden soll.
Data Access (DA)	(*~ (OPC_PROP[0005]:1:Schreibgeschützt) *)	Setzt den Schreibschutz für eine Variable. Muss zusammen mit (*~ (OPC:1: available) *) verwendet werden.
Data Access (DA)	(*~ (OPC_UA_PROP[5100] : x: Alias name) *)	Bestimmt x als den Knotennamen im UA-Namensraum, das sogenannte Alias Mapping.
Historical Access (HA) [▶ 45]	(*~ (OPC_UA_PROP[5000]:x:Storage media) *)	Aktiviert eine Variable für “Historical Access”. Muss zusammen mit (*~ (OPC:1: available) *) verwendet werden. x definiert das Speichermedium, das für die Speicherung der Datenwerte verwendet werden soll, und kann eines der folgenden sein: 1 = Speicher 2 = Datei 3 = SQL Compact Database 4 = SQL Server Database
Historical Access (HA) [▶ 45]	(*~ (OPC_UA_PROP[5000] [1]:x:SamplingRate) *)	Legt die Abtastrate fest, mit der die Variablenwerte zu speichern sind, in Abhängigkeit des Parameters „x“ in [ms]
Historical Access (HA) [▶ 45]	(*~ (OPC_UA_PROP[5000][2]:x:Buffer) *)	Legt die maximale Anzahl im Datenspeicher zu haltender Werte, in Abhängigkeit des Parameters „x“.

4.1.4 C++

4.1.4.1 Zugriff auf C++ Laufzeit

In diesem Teil der Dokumentation wird beschrieben, wie der Namensraum des OPC-UA Servers zu konfigurieren ist, um Zugriff auf die Symbole eines TwinCAT 3 C++ Moduls zu erhalten. Hierzu können Sie entweder den OPC-UA Konfigurator verwenden, oder die Konfiguration direkt und per Hand in der ServerConfig.xml Datei des OPC-UA Servers vornehmen.

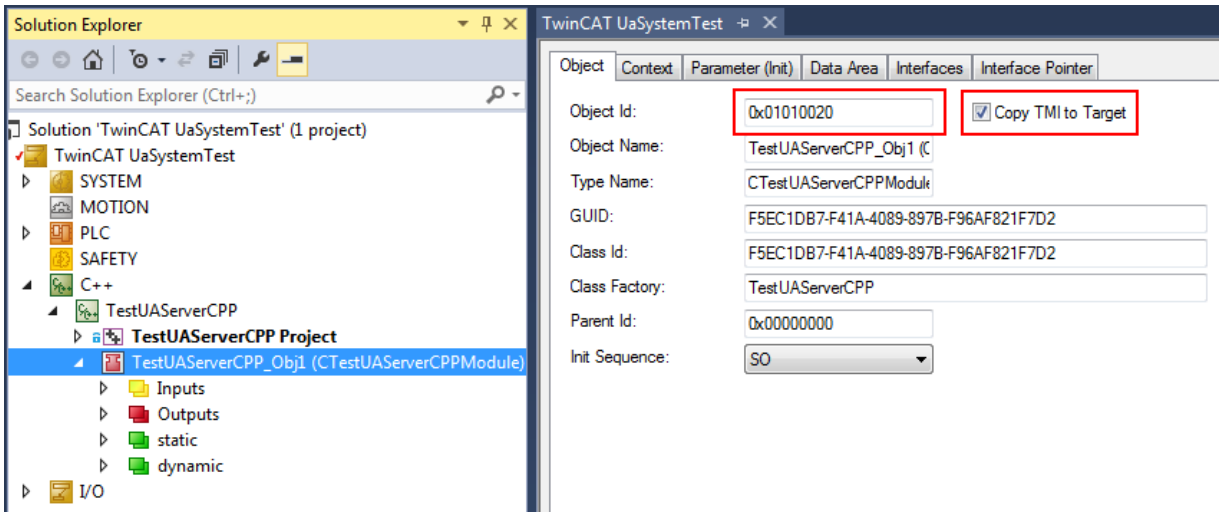
Dieser Artikel beinhaltet folgende Themen:

- Schritt 1: Projektbezogene Einstellungen
- Schritt 2: Konfiguration des UA-Servers

Schritt 1: Projektbezogene Einstellungen

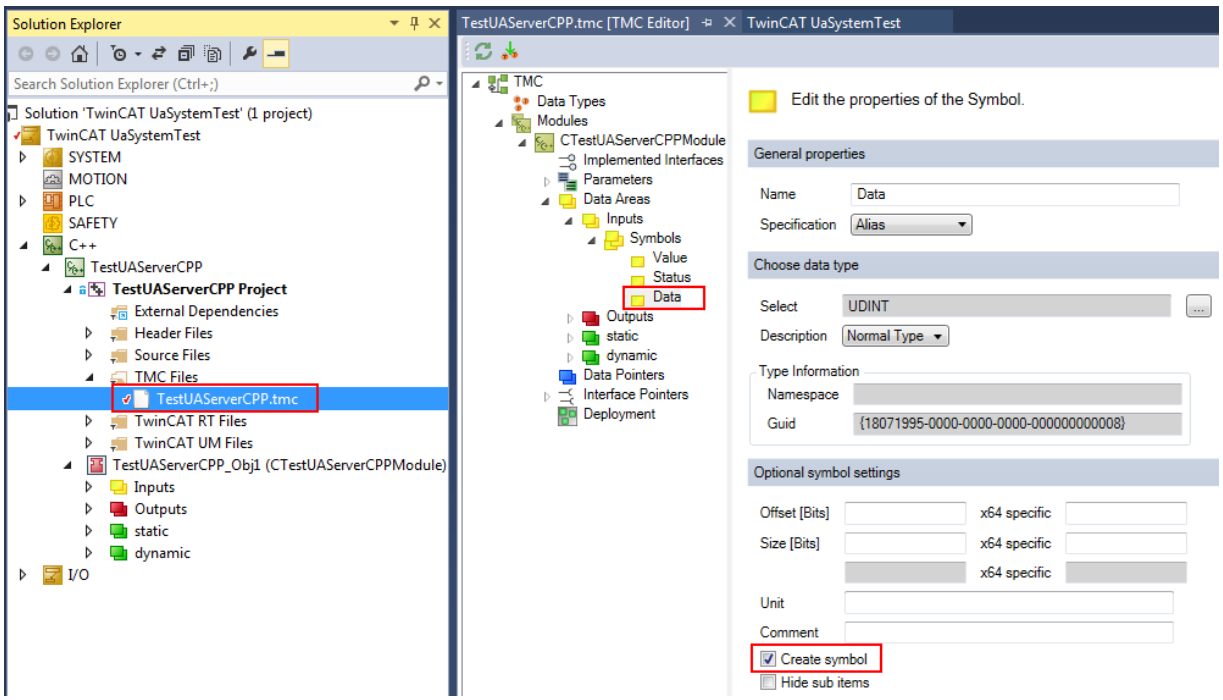
Um bestimmte, in einer Instanz eines TwinCAT 3 C++ Moduls enthaltene Symbole so zu konfigurieren, dass sie über OPC-UA zugänglich werden, sind die folgenden Schritte in den Einstellungen der entsprechenden C ++ Modulinstanz in TwinCAT XAE erforderlich:

1. Der OPC-UA Server benötigt die TMI Symboldatei, die standardmäßig nicht an die Ziel-Laufzeit übergeben wird. Aktivieren Sie die Übertragung durch Setzen der folgenden Option:



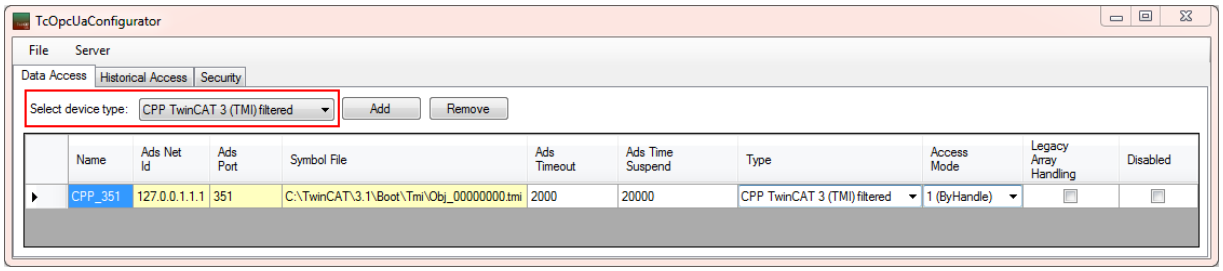
⇨ Die erzeugte TMI-Datei wird nach der ObjectId benannt, z.B. “Obj_01010020.tmi” und in das Twin-CAT Bootverzeichnis, z.B. “C:\TwinCAT\Boot\Tmi\”, abgelegt.

2. Im nächsten Schritt müssen Sie festlegen, welche Symbole für die entsprechenden Variablen zugänglich gemacht werden sollen, indem Sie das Kontrollkästchen “ Create Symbol” im TMC-CodeGenerator aktivieren. Achten Sie darauf, dass Sie den TMC-CodeGenerator anschließend ausführen.



Schritt 2: Konfiguration des UA-Servers

3. Sie können den Zugang zu TwinCAT 3 C++ Modulen bequem mit Hilfe des OPC-UA Konfigurators konfigurieren und parametrisieren. Hierzu müssen Sie ein neues Gerät vom Typ “CPP TwinCAT 3 (TMI) gefiltert” im Bereich “Datenzugriff” hinzufügen.



4. Das Feld “ SymbolFile” muss auf die TMI-Datei zeigen, die für die C++ Modulinstanz erzeugt wurde, siehe obige Schritte. Diese TMI-Dateien befinden sich im TwinCAT Bootverzeichnis, z.B. “C:\TwinCAT \Boot\Tmi\”, und wurden nach der ObjectId der TwinCAT 3 C++ Modulinstanz benannt.

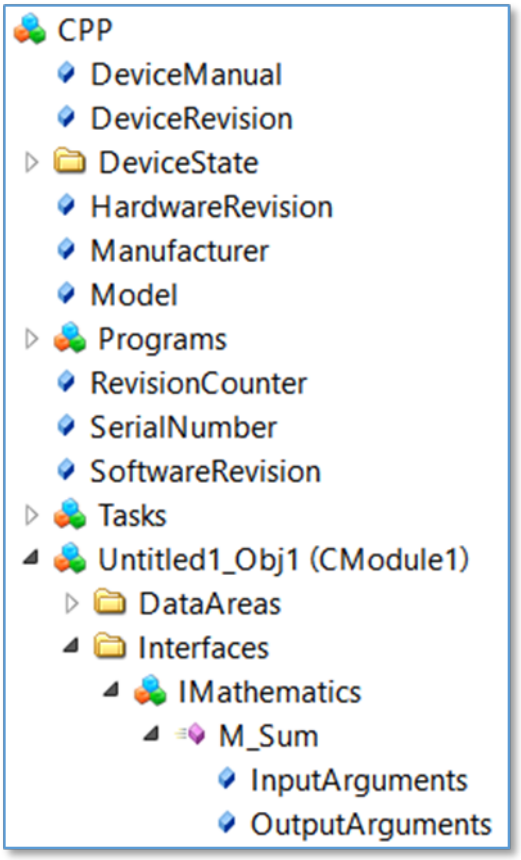
Die einstellbaren Parameter beschreiben die folgenden Funktionen:

Parameter	Beschreibung	Mögliche Werte
ADS Port	Definiert den ADS-Port, unter dem die C++ Modulinstanz zugänglich ist. Der ADS Port kann in den Eigenschaften des C++ Tasks gelesen werden.	351 352 …
AutoCfg	Definiert zunächst den Typ der für die Kommunikation verwendeten Ziel-Laufzeit, z.B. SPS, C++, I/O. Anschließend kann eine weitergehende Unterscheidung innerhalb dieser Kategorien stattfinden. Jede AutoCfg-Option steht sowohl als ungefilterte, als auch gefilterte Option zur Verfügung. Gefiltert bedeutet, dass der Nutzer bestimmen kann, welche C++ Symbole über OPC-UA öffentlich zugänglich gemacht werden. Bei Verwendung einer ungefilterten Option wird jedes C++ Symbol an OPC-UA veröffentlicht.	4020 CPP TwinCAT 3 (TMI) 4021 CPP TwinCAT 3 (TMI) gefiltert
AutoCfgSymFile	Symboldatei (TMI) der entsprechenden C++ Modulinstanz.	Pfad zur Symboldatei. (TMI)
IoMode	Definiert die Methode für den Zugriff auf Symbole.	1 (Zugriff über Handle - standardmäßig)
ArraySubItemLegacySupport	Unterelemente eines Array werden standardmäßig nicht als separate Knoten im UA-Namensraum abgebildet. Stattdessen wird lediglich das Array als ein einzelnes Element abgebildet. Nichtsdestotrotz können UA-Clients über deren sogenannten "IndexRange" auf Unterelemente zugreifen. Unglücklicherweise unterstützen einige ältere OPC-UA Clients diese Möglichkeit noch nicht. Dieses Flag wurde eingeführt, sodass der Zugriff nichtsdestotrotz für diese Clients möglich wird. Das Flag sorgt dafür, dass jede Array-Position als separater Knoten im UA-Namensraum angezeigt wird. Achtung: Dies führt zu einem höheren Speicherbedarf des OPC-UA Servers!	0 (deaktiviert - standardmäßig) 1 (aktiviert)
Disabled	Deaktiviert die C++ Modulinstanz im UA-Namensraum, woraufhin der entsprechende Knoten nicht angezeigt wird. Wir empfehlen die Aktivierung dieses Parameters wenn bestimmte C++ Laufzeiten zum Zeitpunkt der Projektplanung noch nicht verfügbar sind, weil z.B. die entsprechenden Geräte noch nicht ans Netzwerk angeschlossen sind.	0 (deaktiviert - standardmäßig) 1 (aktiviert)

4.1.4.2 Methodenaufruf

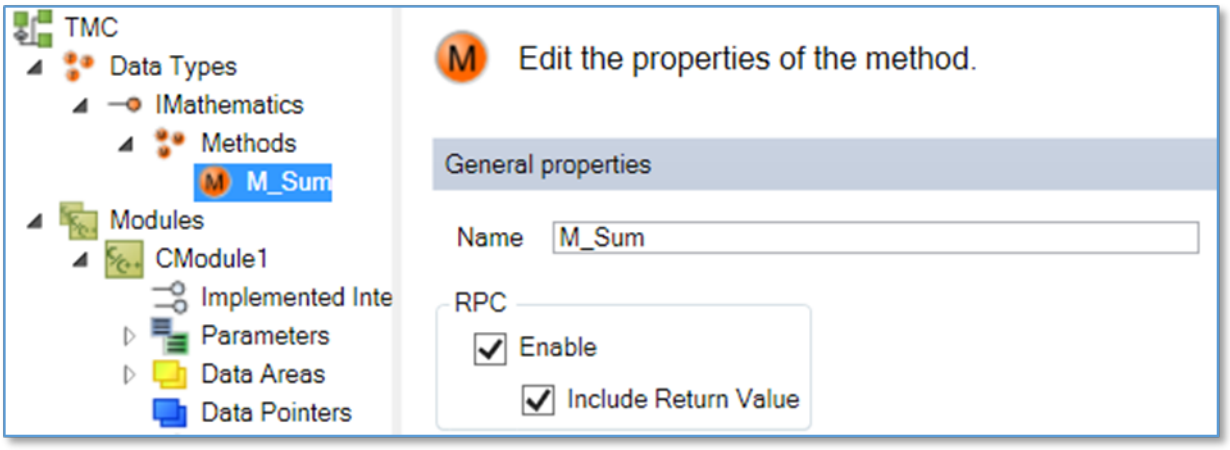
Methodenaufrufe sind ein sehr grundlegender Teil der OPC-UA Spezifikation. Mit der Einführung dieser Funktionalitäten in die SPS-Welt, bietet TwinCAT 3 den Kunden eine sehr effiziente Weise zur Ausführung von RPC-Aufrufen im C++ Echtzeitkontext und verringert somit die klassischen Handshake Patterns bei der Kommunikation zwischen den Geräten. Der TwinCAT OPC-UA Server importiert C++ Methoden wie OPC-UA Methoden über dessen TMI-Import.

Beachten Sie dabei, dass, wenngleich die C++ Methode eine normale Methode im UA-Namensraum zu sein scheint, sie immer noch eine Echtzeit-Methode ist, die innerhalb des Echtzeit-Kontexts ausgeführt wird - und somit unter den Kontext einer Echtzeit-Task fällt! Der TwinCAT C++ Entwickler muss daher Vorsichtsmaßnahmen treffen, dass die Ausführungszeit der Methode in die Task-Zykluszeit passt!

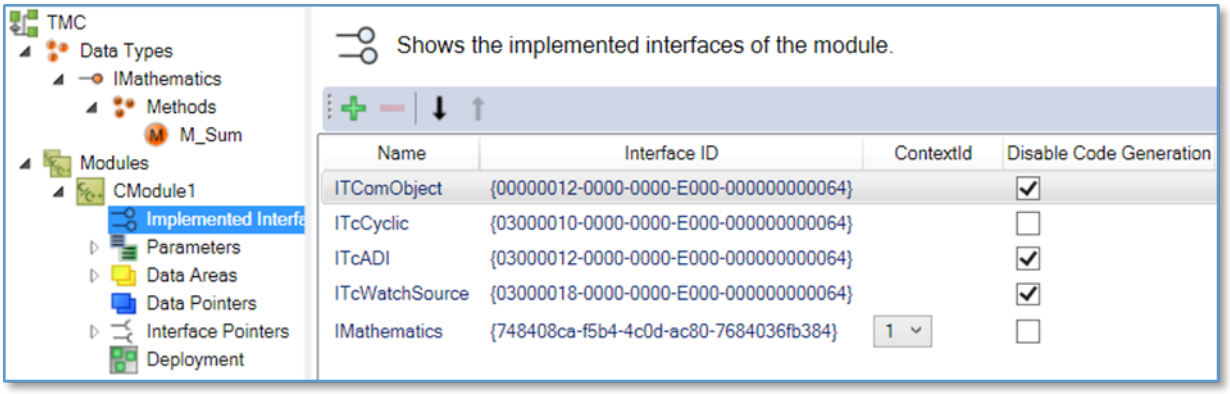


Methoden in TwinCAT 3 C++

TwinCAT Module (TcCOM Module) könnten Schnittstellen implementieren, die über vordefinierte Methoden verfügen. Die Methode selbst muss für RPC-Aufrufe während ihrer Definition aktiviert sein (siehe TwinCAT Module Class Wizard Dokumentation), so dass der OPC-UA Server weiß, dass sie zur Ausführung bereitsteht.

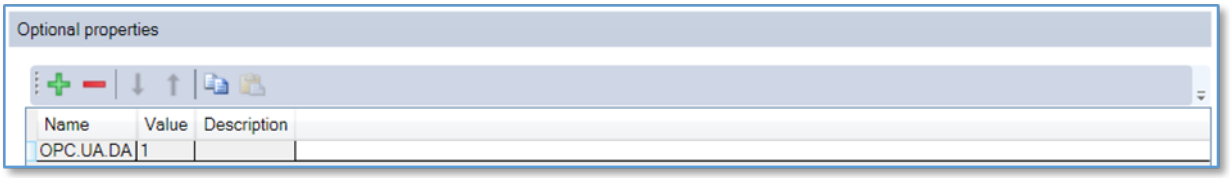


Damit auch der ReturnValue der Methode zur Verfügung steht, muss das entsprechende Feld markiert sein. Denken Sie daran, dass die Schnittstelle, unter der die Methode erstellt wurde, implementiert werden muss.



Gefilterter oder ungefilterter Modus

In Abhängigkeit davon, welcher Importmodus verwendet wird, muss die Methode für den Zugriff über OPC-UA deklariert werden. Dies kann durch Verwendung des TMC Editors und der üblichen OPC-UA Attribute als optionale Eigenschaften erfolgen.



4.1.4.3 Liste der Attribute und Kommentare

Die Konfiguration der Laufzeitvariablen für verschiedene OPC-UA Funktionalitäten (z. B. Data Access oder Historical Access) erfolgt direkt vom SPS-Programm oder vom TMC Code Editor aus (bei Verwendung von TwinCAT 3 C++). Der Vorteil besteht darin, dass ein SPS-Entwickler direkt im Programmcode, mit dem er vertraut ist, entscheiden kann, ob und wie eine Variable für OPC-UA freizugeben ist. Eine derartige Aktivierung wird dann mittels Einfügen eines Kommentars zusammen mit dem entsprechenden OPC-UA-Tag vor der Variablen vorgenommen, z.B.:

TwinCAT 3 SPS (TMC):

```
{attribute 'OPC.UA.DA' := '1'}
bVariable : BOOL;
```

TwinCAT 2 SPS (TPY):

```
bVariable : BOOL; (*~ (OPC:1:available)*)
```

Eine detaillierte Beschreibung darüber, wie Sie Attribute und Kommentare verwenden können, erhalten Sie in dem entsprechenden Kapitel der Laufzeitkomponenten, die verwendet werden (PLC, C++, I/O Task, ...).

In der folgenden Tabelle wird ein Überblick über alle definierbaren Tags und deren Bedeutung gegeben. Siehe jeweiligen Unterabschnitt der entsprechenden Funktion für eine ausführliche Beschreibung des Funktionsprinzips.

Tab. 8: TwinCAT 3 (TMC):

OPC-UA Funktion	SPS Tag	C++ TMC Code Editor (Optionale Eigenschaften)	Bedeutung
Data Access (DA)	{attribute 'OPC.UA.DA' := '0'}	Name: OPC.UA.DA Wert: 0	Sperrt eine Variable für OPC-UA, woraufhin diese im UA-Namensraum nicht mehr sichtbar ist.
Data Access (DA)	{attribute 'OPC.UA.DA' := '1'}	Name: OPC.UA.DA Wert: 1	Aktiviert eine Variable für OPC-UA, woraufhin diese im UA-Namensraum sichtbar wird. Dieser Tag muss immer gesetzt werden, wenn eine Variable für UA verwendet werden soll.
Data Access (DA)	{attribute 'OPC.UA.DA.Access' := 'x'}	Name: OPC.UA.DA.Access Wert: siehe rechte Spalte	Bestimmt den Lese-/Schreib-Zugriff für eine Variable, in Abhängigkeit des Parameters „x“. 0 = Kein 1 = Schreibgeschützt 2 = Nur Schreibzugriff 3 = Lese- und Schreibzugriff (Standardwert, wenn Tag nicht verwendet wird)
Historical Access (HA) [▶ 45]	{attribute 'OPC.UA.HA' := '1'}	Name: OPC.UA.HA Wert: 1	Aktiviert eine Variable für “Historical Access”. Muss zusammen mit {attribute 'OPC.UA.DA' := '1'} verwendet werden.
Historical Access (HA) [▶ 45]	{attribute 'OPC.UA.HA.Storage' := 'x'}	Name: OPC.UA.HA.Storage Wert: siehe rechte Spalte	Definiert den Speicherort für Historical Access, in Abhängigkeit des Parameters “x” 1 = Speicher 2 = Datei 3 = SQL Compact Database 4 = SQL Server Database
Historical Access (HA) [▶ 45]	{attribute 'OPC.UA.HA.Sampling' := 'x'}	Name: OPC.UA.HA.Sampling Wert: siehe rechte Spalte	Legt die Abtastrate fest, mit der die Variablenwerte zu speichern sind, in Abhängigkeit des Parameters „x“ in [ms]
Historical Access (HA) [▶ 45]	{attribute 'OPC.UA.HA.Buffer' := 'x'}	Name: OPC.UA.HA.Buffer Wert: siehe rechte Spalte	Legt die maximale Anzahl im Datenspeicher zu haltender Werte, in Abhängigkeit des Parameters „x“.

Tab. 9: TwinCAT 2 (TPY):

OPC-UA Funktion	SPS Tag	Bedeutung
Data Access (DA)	(*~ (OPC:0:not available) *)	Sperrt eine Variable für OPC-UA, woraufhin diese im UA-Namensraum nicht mehr sichtbar ist.
Data Access (DA)	(*~ (OPC:1:available) *)	Aktiviert eine Variable für OPC-UA, woraufhin diese im UA-Namensraum sichtbar wird. Dieser Tag muss immer gesetzt werden, wenn eine Variable für UA verwendet werden soll.
Data Access (DA)	(*~ (OPC_PROP[0005]:1:Schreibgeschützt) *)	Setzt den Schreibschutz für eine Variable. Muss zusammen mit (*~ (OPC:1: available) *) verwendet werden.
Data Access (DA)	(*~ (OPC_UA_PROP[5100] : x: Alias name) *)	Bestimmt x als den Knotennamen im UA-Namensraum, das sogenannte Alias Mapping.
Historical Access (HA) [▶ 45]	(*~ (OPC_UA_PROP[5000]:x:Storage media) *)	Aktiviert eine Variable für “Historical Access”. Muss zusammen mit (*~ (OPC:1: available) *) verwendet werden. x definiert das Speichermedium, das für die Speicherung der Datenwerte verwendet werden soll, und kann eines der folgenden sein: 1 = Speicher 2 = Datei 3 = SQL Compact Database 4 = SQL Server Database
Historical Access (HA) [▶ 45]	(*~ (OPC_UA_PROP[5000] [1]:x:SamplingRate) *)	Legt die Abtastrate fest, mit der die Variablenwerte zu speichern sind, in Abhängigkeit des Parameters „x“ in [ms]
Historical Access (HA) [▶ 45]	(*~ (OPC_UA_PROP[5000][2]:x:Buffer) *)	Legt die maximale Anzahl im Datenspeicher zu haltender Werte, in Abhängigkeit des Parameters „x“.

4.1.5 I/O

4.1.5.1 Zugriff auf IO-Task

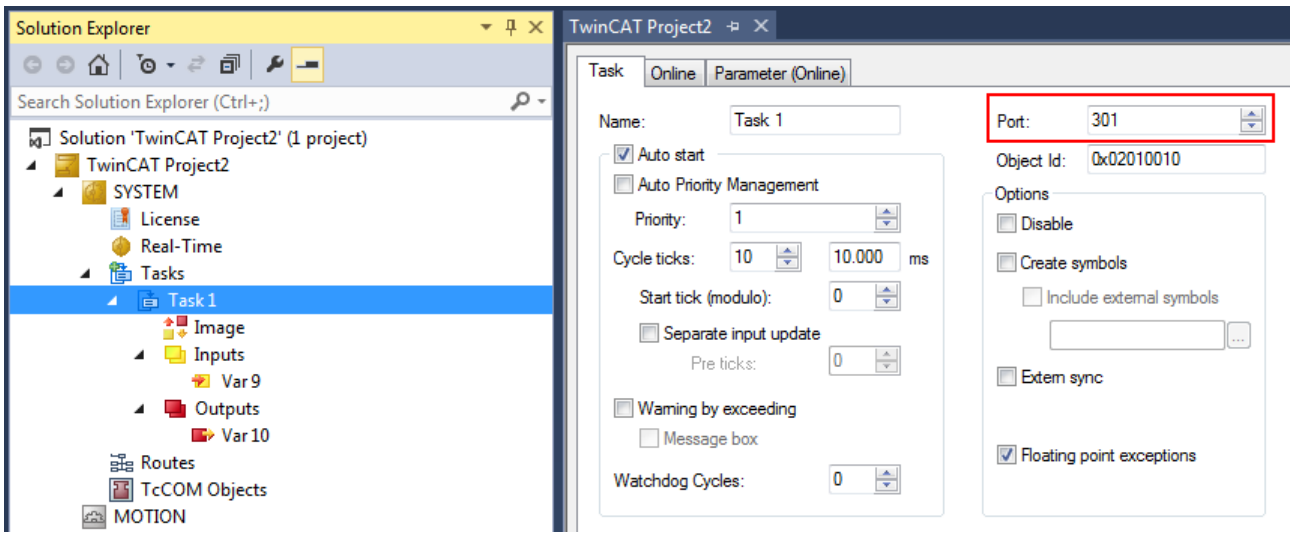
In diesem Artikel wird beschrieben, wie IO Taskvariablen über die Datenzugriffsfunktionen des OPC-UA Servers zugänglich gemacht werden können.

Um IO-Variablen über OPC-UA zugänglich zu machen, müssen zwei Schritte nur einmal konfiguriert werden.

- Schritt 1: Konfiguration des TwinCAT IO Tasks um das Symbolhandling zu aktivieren
- Schritt 2: Konfiguration des OPC-UA Servers um das IO Task in den OPC-UA Namensraum zu integrieren.

Allgemeine Informationen

Die Variablen eines Tasks mit Prozessabbild können über OPC-UA veröffentlicht werden.



Die Art und Weise, wie die Variablen angezeigt und wie mit ihnen kommuniziert wird, ist anhand von mehreren Parametern festgelegt. Diese Parameter können bequem mit Hilfe des OPC-UA Konfigurators festgelegt werden. Alternativ dazu können Sie diese auch direkt in der ServerConfig.xml Konfigurationsdatei festlegen. Die folgende Tabelle gibt einen Überblick über alle Parameter, die beim Zugriff auf das I/O Task von Bedeutung sind.

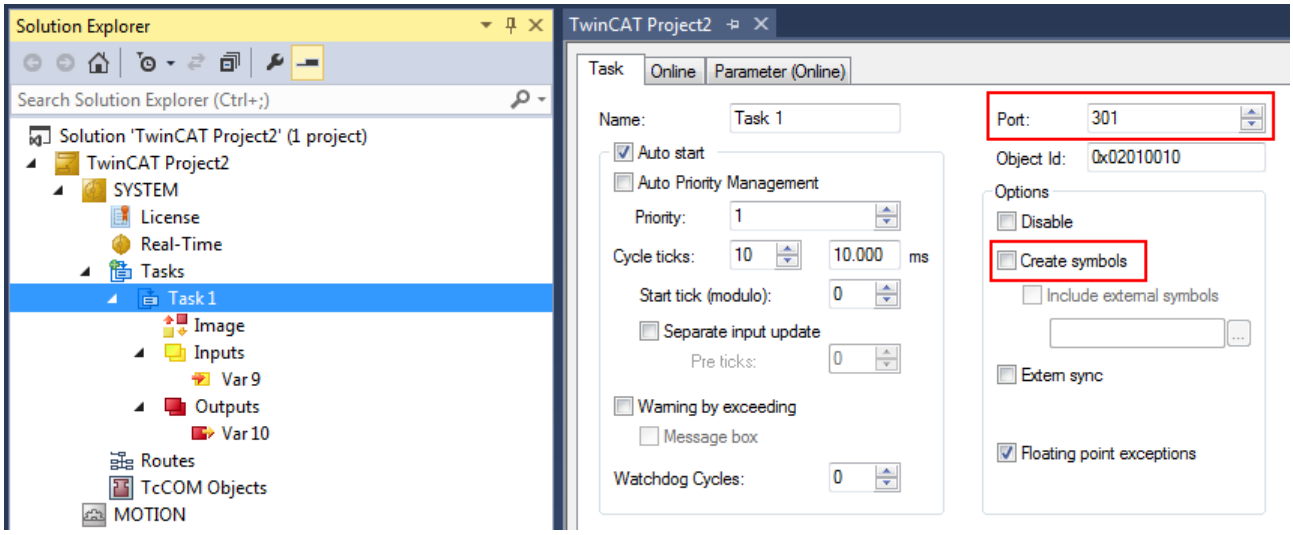
Parameter	Beschreibung	Mögliche Werte
ADS Port	Definiert den ADS-Port, unter dem das I/O Task zugänglich ist. Der ADS Port kann in den Eigenschaften des I/O Tasks ausgelesen werden.	301 302 …
AutoCfg	Definiert zunächst den Typ der für die Kommunikation verwendeten Ziel-Laufzeit, z.B. SPS, C++, I/O. Anschließend kann eine weitergehende Unterscheidung innerhalb dieser Kategorien stattfinden. Jede AutoCfg-Option steht sowohl als ungefilterte, als auch gefilterte Option zur Verfügung. Gefiltert bedeutet, dass die Nutzer bestimmen können, welche I/O Variablen dem OPC-UA über Kommentare öffentlich zugänglich gemacht werden. Bei Verwendung einer ungefilterten Option wird jede Task-Variable an OPC-UA veröffentlicht.	107 I/O TwinCAT 2/3 108 I/O TwinCAT 2/3 gefiltert
AutoCfgSymFile	Bestimmt den Pfad der CurrentConfig.xml Datei, die sich normalerweise im Ordner C:\TwinCAT\3.1\Boot\ befindet.	Pfad von CurrentConfig.xml
Disabled	Deaktiviert die I/O Task im UA-Namensraum, woraufhin der entsprechende Knoten nicht angezeigt wird. Wir empfehlen die Aktivierung dieses Parameters wenn bestimmte Laufzeiten zum Zeitpunkt der Projektplanung noch nicht verfügbar sind, weil z.B. die entsprechenden Geräte noch nicht ans Netzwerk angeschlossen sind.	0 (deaktiviert - standardmäßig) 1 (aktiviert)

Die folgenden Schritte beschreiben, wie Variablen von einem I/O Task in den UA-Namensraum importiert werden können, und welche Schritte hierzu notwendig sind. Es wird vorausgesetzt, dass der OPC-UA Server und die Laufzeit sich auf dem gleichen Rechner befinden.

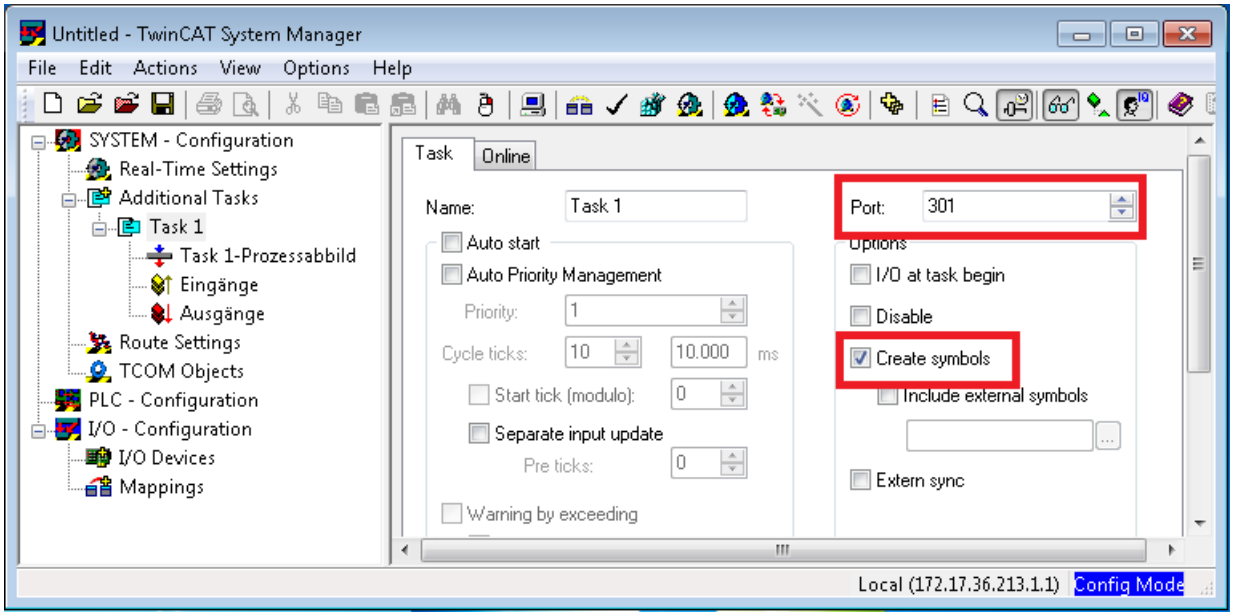
Schritt 1: TwinCAT I/O Task konfigurieren, um das Symbolhandling zu aktivieren.

I/O Task Einstellungen öffnen und die Option “Symbole erzeugen” aktivieren. Notieren Sie an dieser Stelle auch den ADS-Port des I/O Tasks (im Beispiel ist das 301).

TwinCAT 3:

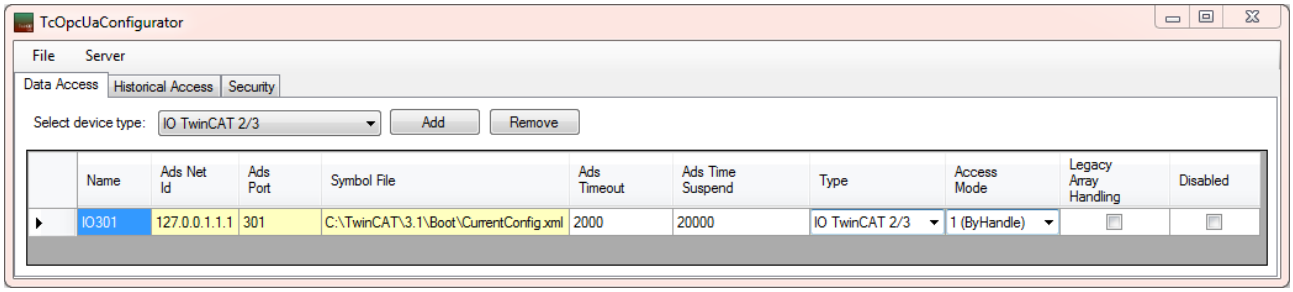


TwinCAT 2:



Schritt 2: Fügen Sie I/O Task dem OPC-UA Namensraum hinzu.

Im nächsten Schritt müssen Sie den OPC-UA Server so konfigurieren, dass er Zugriff auf das I/O Task bietet. Fügen Sie bitte mit Hilfe des OPC-UA Konfigurators ein neues “IO TwinCAT 2/3” Gerät hinzu und konfigurieren Sie dessen Einstellungen für AdsNetId, AdsPort und SymbolFile (Pfad zur CurrentConfig.xml Datei).



Starten Sie, nachdem Sie die entsprechenden Eigenschaften eingestellt haben, den OPC-UA Server neu, damit diese Einstellungen aktiv werden.

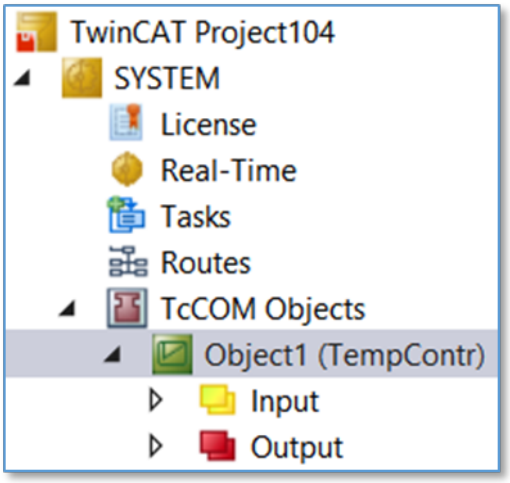
4.1.6 Matlab/Simulink

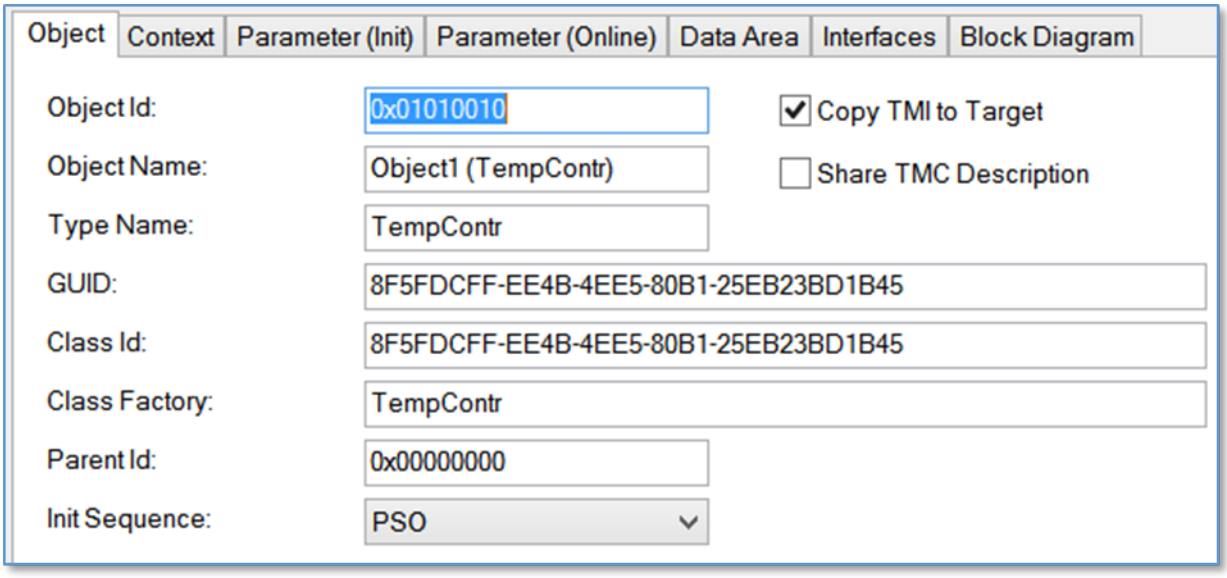
4.1.6.1 Zugriff auf Matlab/Simulink-Module

Der TwinCAT OPC-UA Server bietet die Funktionalität zur Nutzung von TwinCAT 3 TMI-Dateien zum Aufbau der Namensräume. TMI-Dateien werden von TcCOM-Modulen generiert und somit zum Beispiel TwinCAT 3 C++ Modulinstanzen und auch Matlab/Simulink Modulinstanzen.

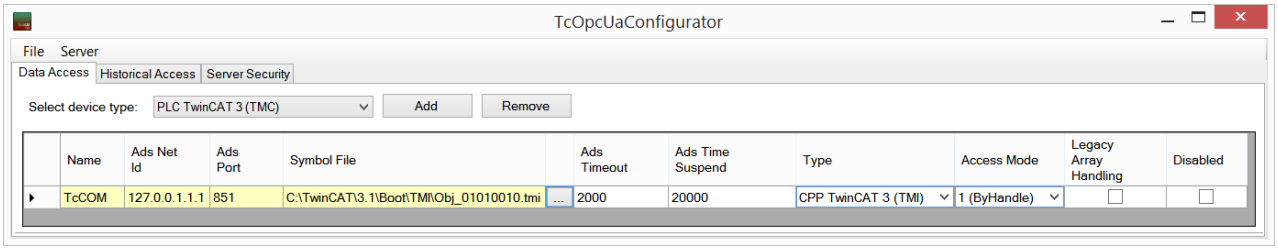
TMI-Import für TcCOM-Module

Der Import von TMI-Dateien kann im TwinCAT OPC-UA Konfigurator konfiguriert werden. Damit eine TMI-Datei generiert und TwinCAT automatisch die TMI-Datei zum Zielsystem kopieren kann, aktivieren Sie bitte die Option „Copy TMI to Target" in den Einstellungen des TcCOM-Moduls:

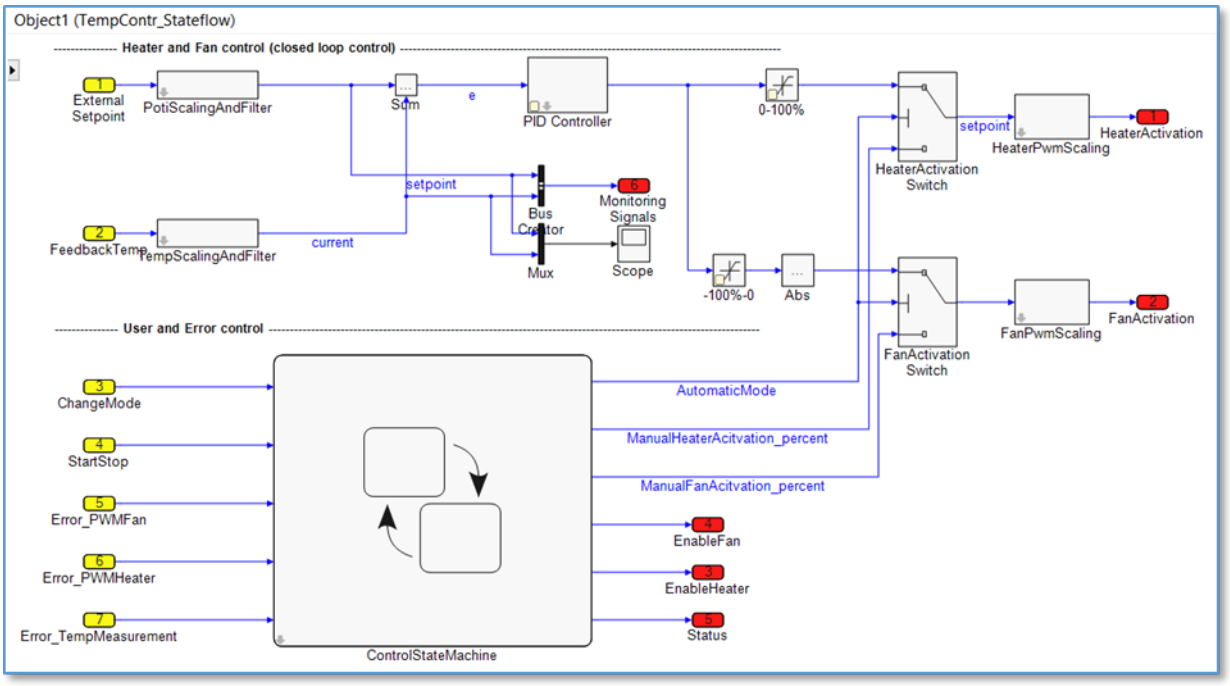


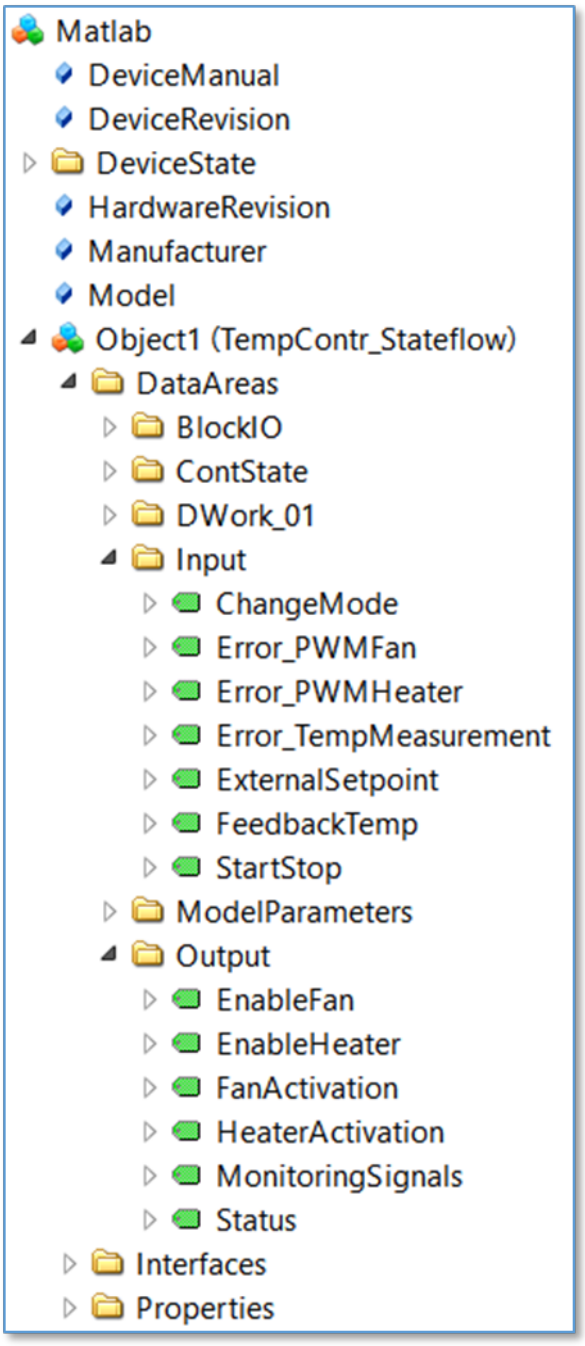


Das Bootverzeichnis des Ziels enthält dann die TMI-Datei für das TcCOM-Modul, welche dann wiederum unter Verwendung des OPC-UA Konfigurators in den OPC-UA Namensraum importiert werden kann:



Der konfigurierte OPC-UA Namensraum enthält dann eine Repräsentation des Matlab/Simulink-Blockdiagramms.





Gefilterter oder ungefilterter Modus

Achten Sie darauf, dass es derzeit keinen gefilterten Modus für generische TcCOM-Modulinstanzen gibt. Der gefilterte Modus ist lediglich für TwinCAT 3 C++ Modulinstanzen möglich, wenn der TMC Code Editor verwendet wird. Die TwinCAT Matlab/Simulink Integration bietet einige Filtermöglichkeiten, die hier beschrieben sind.

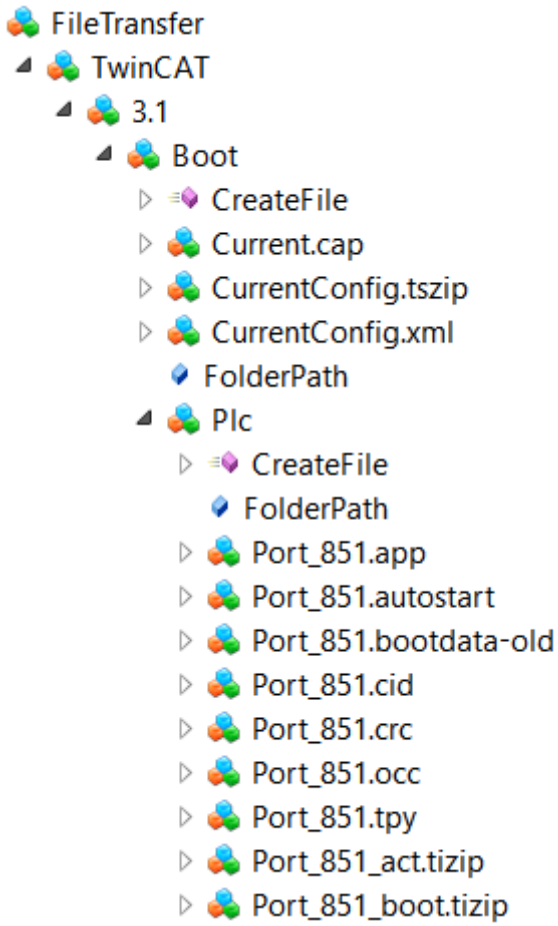
Wenn eine generische TcCOM Modulinstanz (z. B. ein Matlab/Simulink-Modul) importiert werden soll, muss die ungefilterte Option verwendet werden und daher werden alle Symbole der Module im OPC-UA Namensraum veröffentlicht.

4.1.7 Dateiübertragung

4.1.7.1 Auf Dateien und Ordner über OPC-UA zugreifen

Ab OPC-UA Spezifikation Version 1.02 enthält OPC-UA einen spezialisierten ObjectType zur Dateiübertragung, der in Anlage C der Spezifikation beschrieben ist. Dieser spezielle ObjectType, namens FileType, beschreibt das Informationsmuster für die Datenübertragung. Die Dateien können in OPC-UA unter Verwendung von ByteStrings als einfache Variablen modelliert werden. Der FileType repräsentiert eine Datei mit Methoden zum Zugriff auf die Datei. In der OPC-UA Spezifikation erhalten Sie weitere Informationen über den FileType und das Design und die Handhabung der zugrundeliegenden Methoden und Eigenschaften zum Zugriff auf eine Datei im OPC-UA Namensraum.

Beckhoff hat einen generischen Weg implementiert, um Dateien und Ordner von einer lokalen Festplatte in den OPC-UA Namensraum zu laden. Jede Datei wird durch einen FileType repräsentiert und ermöglicht Ihnen somit Lese- und Schreibvorgänge auf diese Datei. Zudem enthält jeder Ordner eine Methode CreateFile(), um neue Dateien auf der Festplatte zu erstellen und einen eigenen FolderPath, um den tatsächlichen Pfad zum Ordner auf dem OPC-UA Server zu bestimmen.



Bitte beachten: Diese Funktionalität wurde dem IPC-Diagnostics OPC-UA Server hinzugefügt. Wenngleich der TwinCAT OPC-UA Server ebenfalls einige Teile der Funktionalität Dateiübertragung bereitstellt, wurde die allgemeine Funktionalität, die eine Offenlegung aller Dateien und Ordner ermöglicht, ausschließlich dem OPC-UA Server hinzugefügt, der Bestandteil des IPC-Diagnostics Produkts ist, das automatisch auf jedem Beckhoff Industrie-PC oder Embedded-PC verfügbar ist.

Konfiguration

FileType Objekte werden in einem separaten Namensraum mit der Bezeichnung „FileTransfer” erstellt. Zur Konfiguration dieses Namensraums und zur Auswahl, welche Dateien und Ordner über OPC-UA verfügbar sein sollen, wird eine XML-Datei verwendet, die in dasselbe Verzeichnis verschoben werden muss, in dem

sich die ausführbare Datei des OPC-UA Servers befindet. Die XML-Datei enthält Informationen über den Ordnerpfad und eine Suchmaske, die definiert, welche Dateien im OPC-UA Namensraum veröffentlicht werden:

```
<Files>
  <FolderObject DisplayName="TwinCAT">
    <FolderObject DisplayName="3.1">
      <FolderObject DisplayName="Boot" Path="c:/TwinCAT/3.1/Boot" Search="*.*" >
        <FolderObject DisplayName="Plc" Path="c:/TwinCAT/3.1/Boot/Plc" Search="*.*" ></FolderObject>
        <FolderObject DisplayName="Tmi" Path="c:/TwinCAT/3.1/Boot/Tmi" Search="*.*" ></FolderObject>
      </FolderObject>
    </FolderObject>
  </FolderObject>
</Files>
```

4.1.8 Sicherheit

4.1.8.1 Übersicht

Sicherheit war eine zentrale Anforderung bei der Entwicklung von OPC-UA. Sie wird in verschiedenen Bereichen adressiert:

- Verschlüsselung
- Integrität
- Authentifizierung

Die Vertraulichkeit der ausgetauschten Information wird durch die Verschlüsselung der ausgetauschten Nachrichten sichergestellt. Dabei werden moderne Kryptographie-Algorithmen verwendet. Um auch den zukünftigen Sicherheitsanforderungen gewachsen zu sein können im nachhinein noch stärkere und modernere Algorithmen einer Anwendung hinzugefügt werden, ohne das Protokoll zu verändern. Entsprechend der Anforderungen der jeweiligen Anwendung können verschiedene Sicherheitsstufen gewählt werden. In einigen Bereichen ist es ausreichend die Nachrichten zu signieren um Änderungen durch Dritte auszuschließen, in anderen Fällen dürfen Daten auch nicht von Dritten gelesen werden und ein zusätzliches Verschlüsseln der Nachrichten ist erforderlich. TF6100 OPC-UA bietet die folgenden Security-Level (Security-Endpunkte) an, mit welchen Clients sich verbinden können:

- None: Keine Sicherheit
- Sign: Signiert Nachrichten
- Sign&Encrypt: Signiert und verschlüsselt Nachrichten

Das Signieren der Nachrichten verhindert das Ändern des Inhalts einer Nachricht durch einen Dritten. So wird verhindert, daß Beispielsweise eine Schreibanweisung zum Öffnen eines Schalters durch einen Dritten verfälscht wird und der Schalter statt dessen geschlossen wird.

OPC-UA Anwendungen identifizieren sich über so genannte Software und Applikationsinstanz-Zertifikate. Mit Hilfe von Software-Zertifikaten ist es möglich bestimmten Client Anwendungen einen erweiterten Zugriff auf Informationen eines OPC-UA Servers zu geben, beispielsweise für das Engineering eines OPC-UA Servers. Durch Applikationsinstanz-Zertifikate kann sichergestellt werden, daß ein OPC-UA Server nur mit vorkonfigurierten Clients kommuniziert. Ein Client kann durch das Applikationsinstanz-Zertifikat des Servers sicherstellen, daß er mit dem richtigen Server spricht (ähnlich der Zertifikate eines Web-Browsers). Die Berücksichtigung dieser Zertifikate ist optional, d. h. ein OPC-UA Server kann auch jedem Client den gleichen Zugriff abhängig der Benutzerrechte geben.

Der TwinCAT OPC-UA Client und TwinCAT OPC-UA Server erzeugen ein selbst-signiertes Zertifikat während des ersten Startvorgangs. Dieses Zertifikat besteht aus einem privaten und einem öffentlichen Schlüssel. Der private Schlüssel wird in dem Verzeichnis \PKI\CA\private und der entsprechende öffentliche Schlüssel unter \PKI\CA\certs abgelegt. Wenn ein beliebiger OPC-UA Client eine gesicherte Verbindung über einen der Security-Endpunkte (Sign, Sign&Encrypt) mit dem Server herstellen will, muss der Client den öffentlichen Schlüssel des OPC-UA Servers kennen. Umgekehrt muss der OPC-UA Server den öffentlichen Schlüssel des Clients kennen. Dieser sogenannte Schlüsselaustausch wird in den folgenden Artikeln beschrieben.

4.1.8.2 Authentifizierung

Zusätzlich zum Zertifikataustausch bietet der TwinCAT OPC-UA Server ebenfalls die Möglichkeit, eine Authentifizierung über Name/Passwort auszuführen. Ein verbindender OPC-UA Client muss daher eine gültige Benutzername-/Passwort-Kombination bereitstellen, damit die Verbindung erfolgreich ist.

Der TwinCAT OPC-UA Server verwende Standard-Windows-Mechanismen für die Validierung der Benutzernamen. Wenn der Computer, auf dem der TwinCAT OPC-UA Server läuft, ein Mitglied einer Windows-Domäne ist (z. B. Active Directory), dann wird die Domäne auf eine gültige Benutzername-/Passwort-Kombination geprüft. Wenn der betreffende Computer ein Einzelgerät ist, dann wird die lokale Benutzerdatenbank von Windows überprüft.

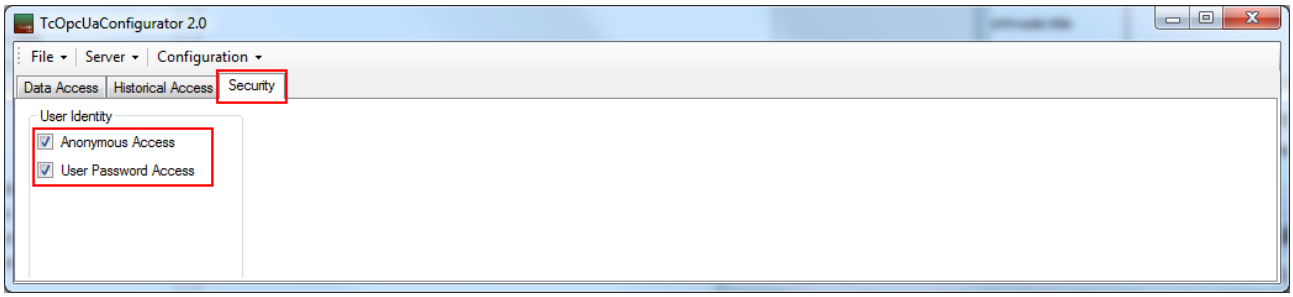
Die Konfiguration des TwinCAT OPC-UA Servers ermöglicht zwei Einstellungen:

- Aktivierung/Deaktivierung anonymer Zugriffe (ohne Authentifizierung)
- Aktivierung/Deaktivierung der Benutzernamen-/Passwort-Validierung

Beide Einstellungen können entweder durch Verwendung des OPC-UA Konfigurators oder manuell durch Bearbeiten der Datei ServerConfig.xml konfiguriert werden.

Konfigurator

Die vorstehenden zwei Einstellungen können über die Registerkarte „Sicherheit“ im OPC-UA Konfigurator konfiguriert werden.



Manuell

Sie können ebenfalls manuell die Einstellungen in der Datei ServerConfig.xml bearbeiten. Zu diesem Zweck müssen Sie die folgenden Komponenten entsprechend anpassen:

```
<UserIdentityTokens>
  <EnableAnonymous>true</EnableAnonymous>
  <EnableUserPw>true</EnableUserPw>
</UserIdentityTokens>
```

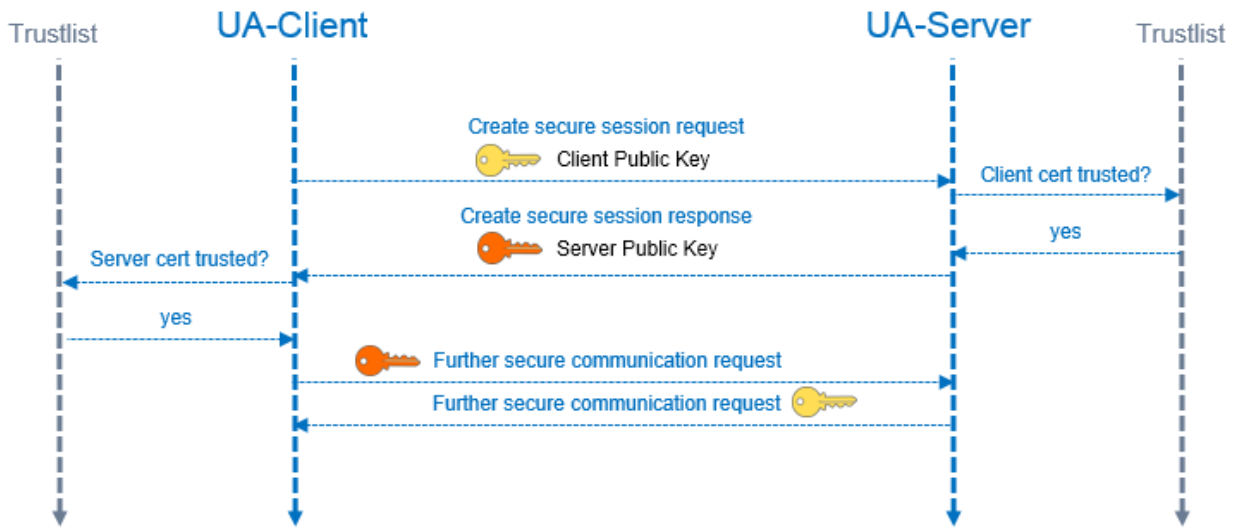
4.1.8.3 Zertifikatsaustausch

Die Kommunikation zwischen einem OPC-UA Client und einem OPC-UA Server kann optional durch Verschlüsselung des Datenverkehrs gesichert werden. Standardmäßig generieren sowohl der TwinCAT OPC-UA Server als auch der TwinCAT OPC-UA Client beim ersten Mal ein maschinenspezifisches, selbstsigniertes Zertifikat zur Authentifizierung.

Wenn Sie in Ihrer Umgebung eine verschlüsselte Verbindung verwenden möchten, müssen Sie ein Vertrauensverhältnis zwischen OPC-UA Server und OPC-UA Client aufbauen.

Dieser Artikel besteht aus den folgenden Kapiteln, die den allgemeinen Austausch von Client- und Server-Zertifikaten beschreiben:

- Anmeldung des OPC-UA Clients beim OPC-UA Server
- Anmeldung des OPC-UA Servers beim OPC-UA Client



Bekanntmachen des OPC-UA Clients beim OPC-UA Server

Möchten Sie einen oder mehrere OPC-UA Clients über Zertifikate am OPC-UA Server authentifizieren, so muss der OPC-UA Server den öffentlichen Schlüsseln der Clients vertrauen. Hierzu muss der öffentliche Schlüssel des jeweiligen Clients in den folgenden Unterordner des OPC-UA Servers kopiert werden:

- TwinCAT 2: **%InstallDir%\UA\PKI\CA\certs**
- TwinCAT 3: **%InstallDir%\Server\PKI\CA\certs**

Dieser Ordner beinhaltet alle Client-Zertifikate, welchen der OPC-UA Server vertraut und mit denen eine authentifizierte Verbindung aufbaut werden kann. Die Client-Zertifikate müssen hierbei im Format **"<beliebigerName>.der"** vorliegen, damit der OPC-UA Server diese verarbeiten kann. Die folgende Anleitung zeigt Ihnen einen Weg, wie man an das Client-Zertifikat gelangt, was für viele Anwendungsfälle wahrscheinlich die einfachste Möglichkeit ist an ein Clientzertifikat zu gelangen. Hierbei bauen Sie, ohne vorher das Clientzertifikat auf den UA-Server kopiert zu haben, eine Verbindung zum UA-Server unter Verwendung eines Security-Endpunkts (z.B. Basic128Rsa15/Sign&Encrypt) auf. Diese Verbindung wird vom UA-Server natürlich abgewiesen, da dieser dem UA-Client zum derzeitigen Zeitpunkt noch nicht vertraut. Der TwinCAT OPC-UA Client würde in diesem Fall z.B. den Fehler 0xE4DD0102 zurückliefern. Nach Abweisen der Verbindungsaufforderung speichert der UA-Server jedoch eine Kopie des Clientzertifikats im folgenden Verzeichnis ab:

- TwinCAT 2: **%InstallDir%\UA\PKI\CA\crl**
- TwinCAT 3: **%InstallDir%\Server\PKI\CA\crl**

Der Name des Zertifikats entspricht hierbei dem "Thumbprint" des Zertifikats und lässt sich somit eindeutig dem UA-Clientzertifikat zuordnen. Diese Datei können Sie nun direkt in den "certs"-Ordner kopieren, damit der UA-Server bei weiteren Verbindungsanforderungen dem UA-Client vertraut und die Verbindung akzeptiert.

- TwinCAT 2: **%InstallDir%\UA\PKI\CA\certs**
- TwinCAT 3: **%InstallDir%\Server\PKI\CA\certs**

Bekanntmachen des OPC-UA Servers beim OPC-UA Client

Abhängig vom verwendeten OPC-UA Client müssen eventuell unterschiedliche Schritte unternommen werden, damit der OPC-UA Client dem OPC-UA Server vertraut. Die folgende Anleitung ist daher nur für den TwinCAT OPC-UA Client gültig.

Der öffentliche Schlüssel des OPC-UA Servers befindet sich als DER-Datei in dem folgenden Verzeichnis:

- TwinCAT 2: **%InstallDir%\UA\PKI\CA\certs**
- TwinCAT 3: **%InstallDir%\Server\PKI\CA\certs**

Der Name der Datei lautet **"Beckhoff_OpcUaServer.der"**. Diese Datei muss beim OPC-UA Client in den folgenden Ordner kopiert werden:

- TwinCAT 2: **%InstallDir%\UA Client\PKI\CA\certs**
- TwinCAT 3: **%InstallDir%\Client\PKI\CA\certs**

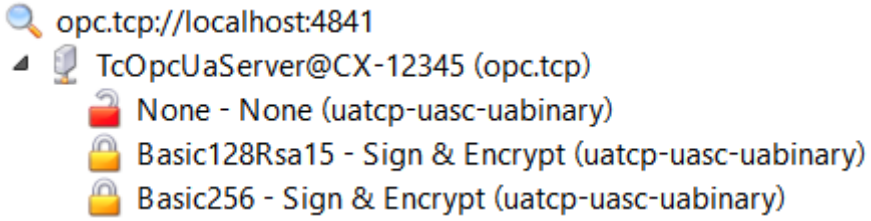
Bei einem Drittanbieter OPC-UA Client muss diese Datei in ähnlicher Form bekannt gemacht werden.

Nun vertraut der entsprechende TwinCAT OPC-UA Client dem UA Server und erlaubt es, eine Verbindung zu diesem herzustellen.

4.1.9 Verschiedenes

4.1.9.1 Endpunkte und Standard-Port

Standardmäßig bietet der TwinCAT OPC-UA Server die folgenden Endpunkte und Standard-Port für OPC-UA Clients zur Verbindung an:



Hinweis

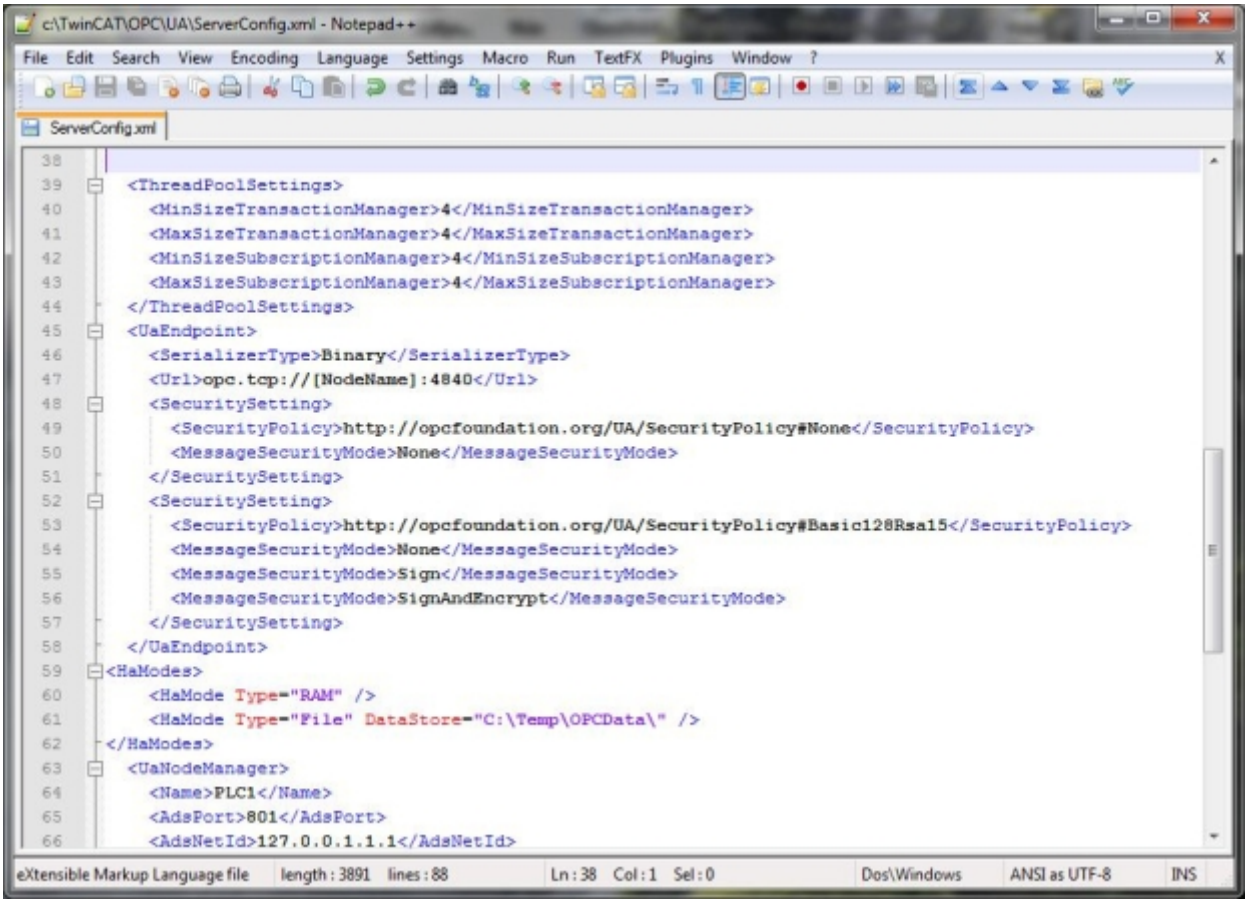
Standard-Port

Beachten Sie bitte, dass der Standard-Port 4840 eventuell von anderen OPC-UA Servern verwendet wird, z. B. dem Local Discovery Server (LDS) von der OPC Foundation, die von manchen Anbietern mit OPC-UA Softwarepaketen eingesetzt wird.

Endpunkt	Beschreibung
Keine - Keine	Keine Sicherheit
Basic128Rsa15 - Sign & Encrypt	Signatur und Verschlüsselung von Nachrichten für mittlere Sicherheitsanforderungen. Erfordert Zertifikataustausch und die Verwaltung der Vertrauenslisten. Weitere Informationen erhalten Sie im OPC-UA Server Sicherheitskapitel [▶ 75].
Basic256 - Sign & Encrypt	Signatur und Verschlüsselung von Nachrichten für mittlere bis hohe Sicherheitsanforderungen. Erfordert mehr Rechenzeit (höhere Anforderungen an die CPU des Computers). Erfordert Zertifikataustausch und die Verwaltung der Vertrauenslisten. Weitere Informationen erhalten Sie im OPC-UA Server Sicherheitskapitel [▶ 75].

4.1.9.2 Konfigurationsdatei

Alle Einstellungen für den TwinCAT OPC-UA Server können mithilfe eines Konfigurators [▶ 103] vorgenommen werden, der über das Setup vom UA-Server Version 1.6.80 geliefert wird. Der Konfigurator macht es für den Benutzer einfacher, den UA-Server über seine „**ServerConfig.xml**" Konfigurationsdatei zu konfigurieren. Die Konfigurationsdatei enthält eine hierarchische, XML-basierte Struktur aller einstellbaren Parameter des UA-Servers. Die folgende Abbildung gibt eine Beispielstruktur dieser Datei wieder.



Wir empfehlen im Allgemeinen die Verwendung des OPC-UA Konfigurators, da auf diese Weise Fehler bei der Konfiguration vermieden werden können.

Die Konfigurationsdatei befindet sich immer in demselben Verzeichnis wie TcOpcUaServer.exe, zum Beispiel:

- Windows XP / 7 mit TwinCAT 2: **%TwinCATDIR%\OPC\UA**
- Windows XP / 7 mit TwinCAT 3: **%TwinCATDIR%\Functions\TF6100-OPC-UA\Win32\Server**
- Windows CE TwinCAT 2 : **HardDisk\System**
- Windows CE TwinCAT 3 : **HardDisk\TwinCAT\Functions\TF6100-OPC-UA\Server**

4.1.9.3 Konfiguration mehrerer TwinCAT Laufzeiten

Sie können optional weitere TwinCAT ADS-Geräte zum UA-Namensraum hinzufügen. Durch die Konfiguration des Namensraums weiß der OPC-UA Server welches TwinCAT-ADS Gerät (z.B. PLC, IO-Task, C++ Laufzeit,...) über den OPC-UA Server verfügbar gemacht werden soll und blendet entsprechend dessen Symbolinformationen im UA-Namensraum ein. Diese Konfiguration muss nur einmalig durchgeführt werden.

Hinzufügen / Entfernen von weiteren ADS-Geräten zum UA-Namensraum

Der TwinCAT OPC-UA Server ist per default so konfiguriert, dass er sich zur ersten SPS-Laufzeit des lokalen Systems verbindet. Um weitere ADS-Geräte bzw. weitere Laufzeiten über den UA-Namensraum verfügbar zu machen, müssen Sie die folgenden Schritte durchführen:

Falls es sich bei dem hinzuzufügenden Gerät um ein lokales Gerät handelt:

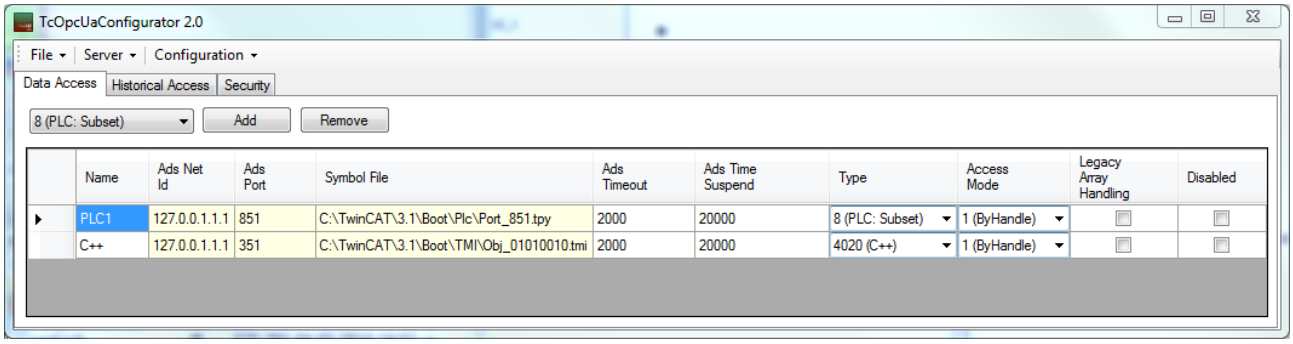
- Hinzufügen des Remotegeräts zur Konfiguration des OPC-UA Servers, z.B. über den OPC-UA Konfigurator oder durch direktes editieren der ServerConfig.xml

Falls es sich bei dem hinzuzufügenden Gerät um ein Remotegerät handelt:

- Herstellen von ADS-Routen zwischen dem Rechner auf dem sich der OPC-UA Server befindet und dem Remotesystem
- Hinzufügen des Remotegeräts zur Konfiguration des OPC-UA Servers, z.B. über den OPC-UA Konfigurator oder durch direktes editieren der ServerConfig.xml

Verwenden des OPC-UA Konfigurators

Verwenden Sie den OPC-UA Konfigurator, um weitere ADS-Geräte zum UA-Namensraum hinzuzufügen.



Im Bereich „Data Access“ des OPC-UA Konfigurators können Sie die im UA-Namensraum eingebundenen ADS-Geräte verwalten bzw. neue Geräte hinzufügen oder existierende Geräte entfernen. Zum Hinzufügen wählen Sie hierzu zunächst den Typ des Geräts aus der zugehörigen DropDown-Box aus (z.B. „8 (PLC: Subset)“) und klicken Sie dann auf „Add“. Es wird ein zusätzlicher Eintrag zur Liste der ADS-Geräte hinzugefügt. Nehmen Sie die gewünschten Einstellungen vor und klicken Sie anschliessend im Menü „Configuration“ auf „Activate“, wenn Sie die Konfiguration sofort aktivieren möchten.

Abhängig vom hinzuzufügenden Gerät müssen unterschiedliche Einstellungen für die einzelnen Parameter vorgenommen werden. Konsultieren Sie hierzu bitte die Dokumentationsartikel aus dem Bereich „Data Access“.

Bitte beachten Sie, dass Sie den OPC-UA Server neu starten müssen, damit die Änderungen wirksam werden. Dies können Sie ebenfalls über den Konfigurator erledigen, indem Sie im Menü „Server“ die Option „Restart“ auswählen. Alternativ reicht hierzu auch ein TwinCAT Neustart.

[Optional] Editieren der ServerConfig.xml

Sie können auch die ServerConfig.xml direkt editieren, um weitere ADS-Geräte zum UA-Namensraum hinzuzufügen. In dieser Konfigurationsdatei gibt es einen Bereich <UaNodeManager>, welcher jeweils den Zugriff auf ein ADS-Gerät definiert, zum Beispiel:

```
    </SecurityPolicy>
    <SecuritySetting>
      <SecurityPolicy>http://opcfoundation.org/UA/SecurityPolicy#Basic128Rsa1
      <MessageSecurityMode>None</MessageSecurityMode>
      <MessageSecurityMode>Sign</MessageSecurityMode>
      <MessageSecurityMode>SignAndEncrypt</MessageSecurityMode>
    </SecuritySetting>
  </UaEndpoint>
  <UaNodeManager>
        <Name>PLC1</Name>
        <AdsPort>801</AdsPort>
        <AdsNetId>127.0.0.1.1.1</AdsNetId>
        <AdsTimeout>2000</AdsTimeout>
        <AdsTimeSuspend>20000</AdsTimeSuspend>
        <AutoCfg>8</AutoCfg>
        <AutoCfgSymFile>C:\TwinCAT\BOOT\CurrentPlc_1.tpy</AutoCfgSymFile>
        <Disabled>0</Disabled>
  </UaNodeManager>
  <UaNodeManager>
        <Name>PLC2</Name>
        <AdsPort>811</AdsPort>
        <AdsNetId>127.0.0.1.1.1</AdsNetId>
        <AdsTimeout>2000</AdsTimeout>
        <AdsTimeSuspend>20000</AdsTimeSuspend>
        <AutoCfg>8</AutoCfg>
        <AutoCfgSymFile>C:\TwinCAT\BOOT\CurrentPlc_2.tpy</AutoCfgSymFile>
        <Disabled>0</Disabled>
  </UaNodeManager>
  <UaNodeManager>
        <Name>IO_Task301</Name>
        <AdsPort>301</AdsPort>
        <AdsNetId>127.0.0.1.1.1</AdsNetId>
        <AdsTimeout>2000</AdsTimeout>
        <AdsTimeSuspend>20000</AdsTimeSuspend>
        <AutoCfg>5</AutoCfg>
        <Disabled>0</Disabled>
  </UaNodeManager>
</UaServerConfig>
<General>
```

Abhängig vom hinzuzufügenden Gerät müssen unterschiedliche Einstellungen für die einzelnen Parameter vorgenommen werden. Konsultieren Sie hierzu bitte die Dokumentationsartikel aus dem Bereich „Data Access“.

Bitte beachten Sie, dass Sie den OPC-UA Server neu starten müssen, damit die Änderungen wirksam werden. Dies können Sie ebenfalls über den Konfigurator erledigen, indem Sie im Menü „Server“ die Option „Restart“ auswählen. Alternativ reicht hierzu auch ein TwinCAT Neustart.

4.1.9.4 Konfiguration mehrerer Server-Instanzen

Es besteht die Möglichkeit, mehrere TwinCAT OPC-UA Server auf einem System zu installieren. Dieser Teil der Dokumentation beschreibt die hierfür notwendigen Konfigurationsschritte. Die Konfiguration ist unter Windows CE und Windows XP basierten Betriebssystemen identisch und besteht aus den folgenden Teilschritten:

- Installation des TwinCAT OPC-UA Servers
- Kopieren der installierten UA-Server Dateien in ein neues Verzeichnis
- Anpassen der Konfigurationsdatei des zweiten UA-Servers

Lediglich die Pfade bei den beiden Betriebssystemvarianten sind unterschiedlich und es wird in den einzelnen Schritten entsprechend auf die Unterschiede hingewiesen.

Hinweis: Der TwinCAT OPC-UA Server ermöglicht eine Nutzung von OPC-UA direkt aus der SPS-Ebene heraus. Hierfür wurde ein ADS-Server in das Produkt integriert, damit SPS-Programmierer per Funktionsbaustein Daten über OPC-UA verschicken können. Möchten Sie, wie in dieser Anleitung beschrieben, mehrere TwinCAT OPC-UA Server parallel auf einem System laufen lassen, so gilt zu beachten, dass nur die **erste** Instanz des UA-Servers per Funktionsbaustein angesprochen werden kann.

Installation des TwinCAT OPC-UA Servers

Die Erst-Installation des TwinCAT OPC-UA Servers erfolgt über die auf dem Beckhoff FTP-Server erhältliche Setup-Datei. Standardmäßig wird der UA-Server in dem folgenden Verzeichnis installiert:

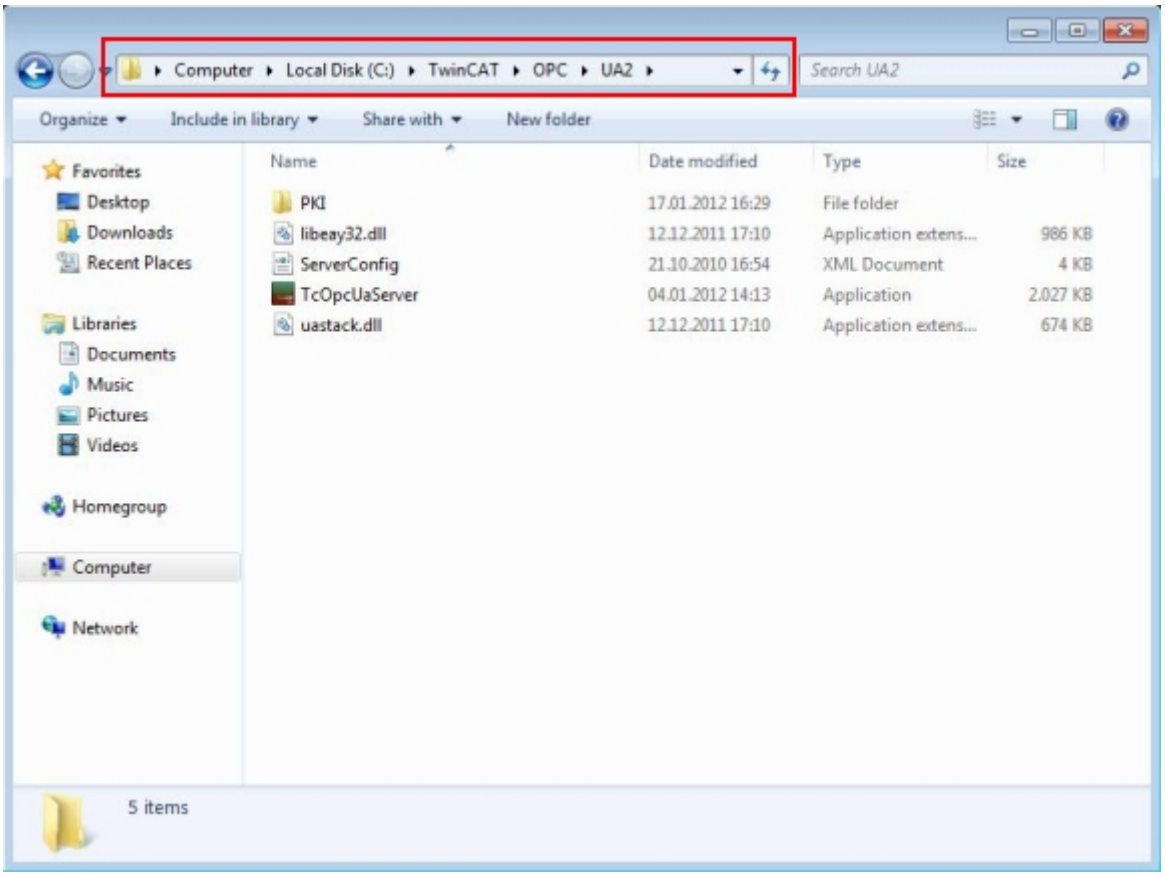
- **Windows XP:** %TwinCATDIR%\Function\TF6100-OPC-UA\Win32\Server\
- **Windows CE:** HDD\System

Kopie der installierten UA-Server Dateien

Im zweiten Schritt müssen die installierten UA-Server Dateien in ein neues Verzeichnis kopiert werden. Erstellen Sie ein neues Verzeichnis namens "Server2" in einem Ordner Ihrer Wahl, z.B.

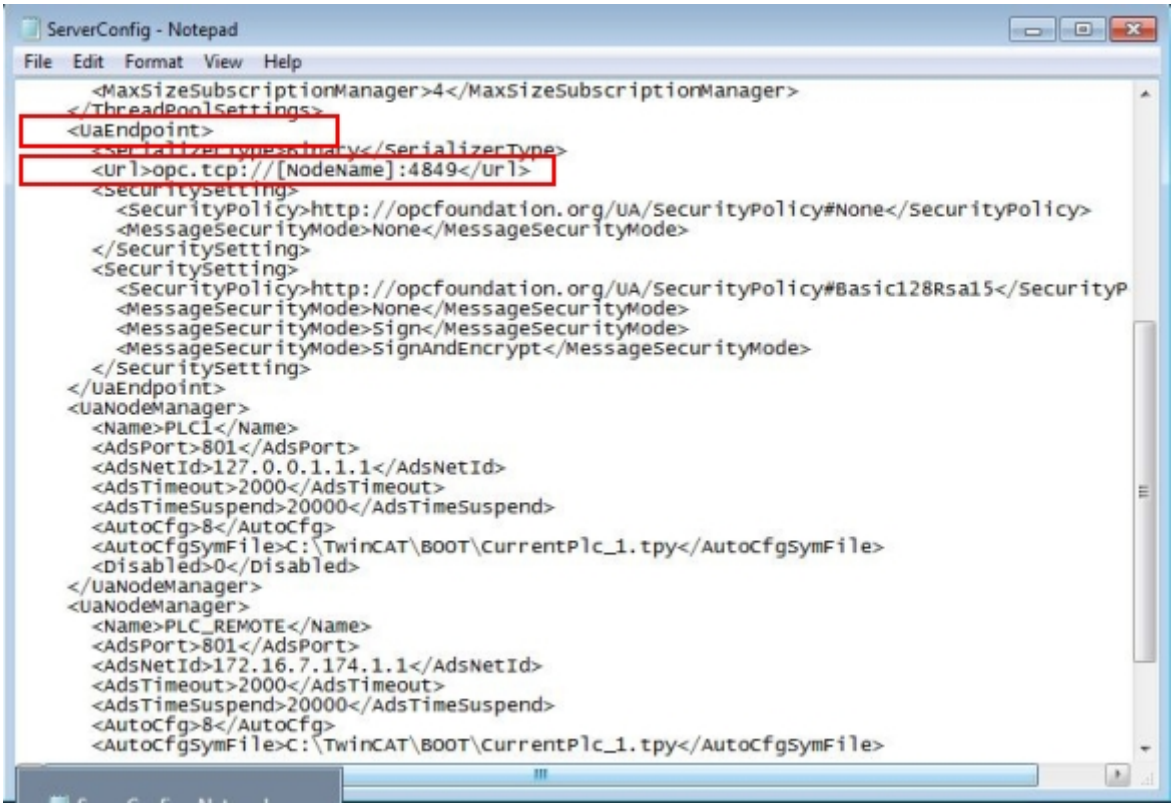
- **Windows XP:** %TWINCATDIR%\Function\TF6100-OPC-UA\Win32\Server2\
- **Windows CE:** HDD\System\UA2

Als nächstes kopieren Sie die Inhalte des Ursprungsordners in das neu erstellte Verzeichnis. Der Inhalt dieses Ordners sollte dann wie folgt aussehen:

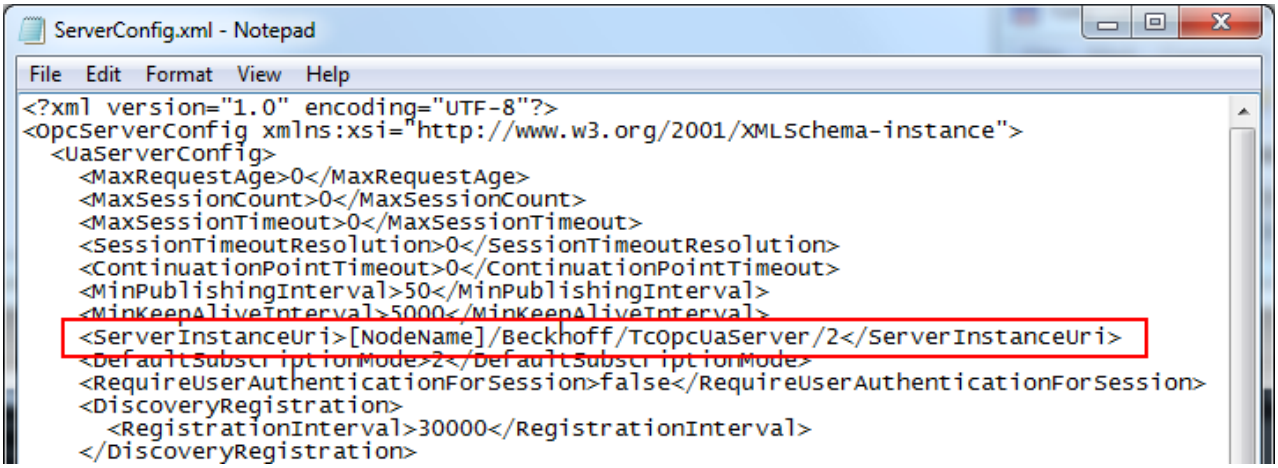


Anpassen der Konfigurationsdatei des zweiten UA-Servers

Damit der zweite UA-Server einen anderen TCP-Port benutzt, muss einmalig dessen Konfigurationsdatei angepasst werden. Hierzu öffnen Sie die Datei "ServerConfig.xml" mit einem Texteditor Ihrer Wahl, zum Beispiel Notepad. Weitere Informationen zur Konfigurationsdatei des UA-Servers finden Sie hier [▶ 78]. Suchen Sie nun in dieser Datei den Eintrag **"<Url>opc.tcp://[NodeName]:4840</Url>"** unterhalb des Knotens **"<UaEndpoint>"** und ändern Sie dessen Portnummer (4840) in einen TCP-Port Ihrer Wahl, z.B. 4849:



Zusätzlich müssen Sie noch den Knoten **<ServerInstanceUri>[NodeName]/Beckhoff/TcOpcUaServer/ 1 </ServerInstanceUri>** anpassen und eine neue Nummer vergeben, zum Beispiel:

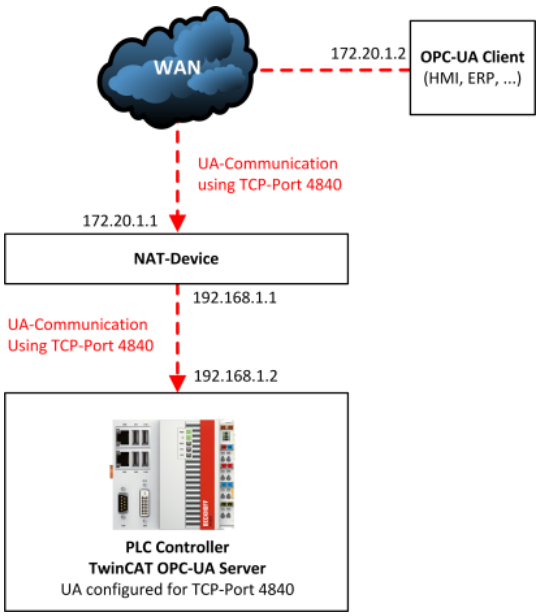


Durch Ausführen der Datei können Sie nun den zweiten TwinCAT OPC-UA Server starten:

- **Windows XP**: %TWINCATDIR%\Function\TF6100-OPC-UA\Win32\Server2\TcOpcUaServer.exe -console

4.1.9.5 Konfiguration von Firewalls

Um eine OPC-UA Kommunikation auch über ein NAT-Gerät, z.B. einen Internet-Router, zu ermöglichen, muss dieses entsprechend den verwendeten UA-Port an den TwinCAT OPC-UA Server weiterleiten können (sogenanntes Port Forwarding). Standardmäßig wird der TwinCAT OPC-UA Server für eine UA-Kommunikation über den TCP-Port 4840 konfiguriert, diese Konfiguration kann jedoch bei Bedarf selbst angepasst werden; entweder über die Server-Konfigurationsdatei [▶ 78] oder den OPC-UA Konfigurator [▶ 103]. Das folgende Schaubild verdeutlicht noch einmal den Zusammenhang von Port Forwarding und dem UA-Server.

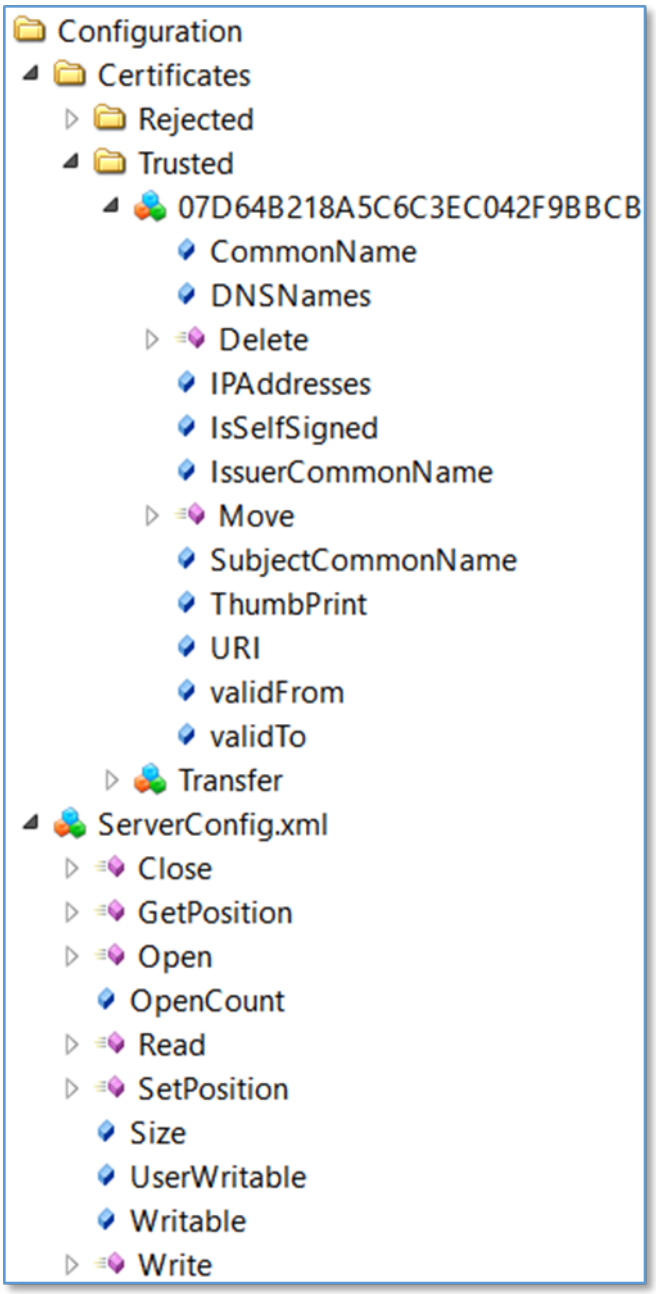


In diesem Beispiel baut der OPC-UA Client eine UA-Verbindung über den TCP-Port 4840 zum NAT-Device auf, welches dann diese Kommunikationsverbindung über ein Port Forwarding an den TwinCAT OPC-UA Server weiterleitet. Das NAT-Gerät muss also nur den im TwinCAT OPC-UA Server konfigurierten UA-Port an das Zielgerät weiterleiten. Den vom UA-Server verwendeten Port können Sie entweder in der Server-Konfigurationsdatei [▶ 78] einsehen oder auch ganz bequem über den UA-Konfigurator [▶ 103], im Auslieferungszustand ist dies jedoch immer der TCP-Port 4840. Die entsprechende Konfiguration Ihres NAT-Geräts für ein Port Forwarding entnehmen Sie bitte dessen Dokumentation.

4.1.9.6 Konfiguration des Namensraums

Ab der Version 2.1.x bietet der TwinCAT OPC-UA Server eine separate Konfiguration des Namensraums, die die folgenden Funktionalitäten enthält:

- Verwaltung der ServerConfig (Lesen/Schreiben-Zugriff) des OPC-UA Servers
- Zertifikatmanagement für vertrauenswürdige/abgelehnte Client-Zertifikate



Verwaltung der ServerConfig

Die ServerConfig wird im Namensraum als regulärer OPC-UA FileType veröffentlicht und bietet daher die entsprechenden Methoden und Eigenschaften, um Zugriff auf die Datei zu erhalten.

Verwaltung von Client-Zertifikaten

Jedes Client-Zertifikat, das dem Server bekannt ist, wird im Namensraum als OPC-UA Zertifikattyp veröffentlicht. Zertifikate werden in „abgelehnte" und „vertrauenswürdige" Zertifikate unterteilt, was durch einen separaten Ordner im Namensraum repräsentiert wird.

Durch Aufruf der Methode Move() kann ein Zertifikat zwischen den Vertrauenslisten verschoben werden.

Zudem bieten verschiedene Eigenschaften zur einfacheren Identifikation weitere Informationen über die Zertifikate selbst.

Verwendung der Namensraum-Konfiguration

Die Namensraum-Konfiguration darf nur von authentifizierten Benutzern verwendet werden. Dies bedeutet, dass ein OPC-UA Client sich selbst gegenüber dem OPC-UA Server durch Bereitstellung einer gültigen Benutzername-/Passwort-Kombination authentifizieren muss. Gültige Benutzernamen sind entweder lokale Benutzer oder auch Benutzer von einer Windows-Domäne, wenn der Computer, auf dem der OPC-UA Server läuft, ein Mitglied einer Windows-Domäne ist.

In seiner ServerConfig stellt der OPC-UA Server die folgenden XML-Tags bereit, um den Zugriff auf die Namensraum-Konfiguration zu konfigurieren:

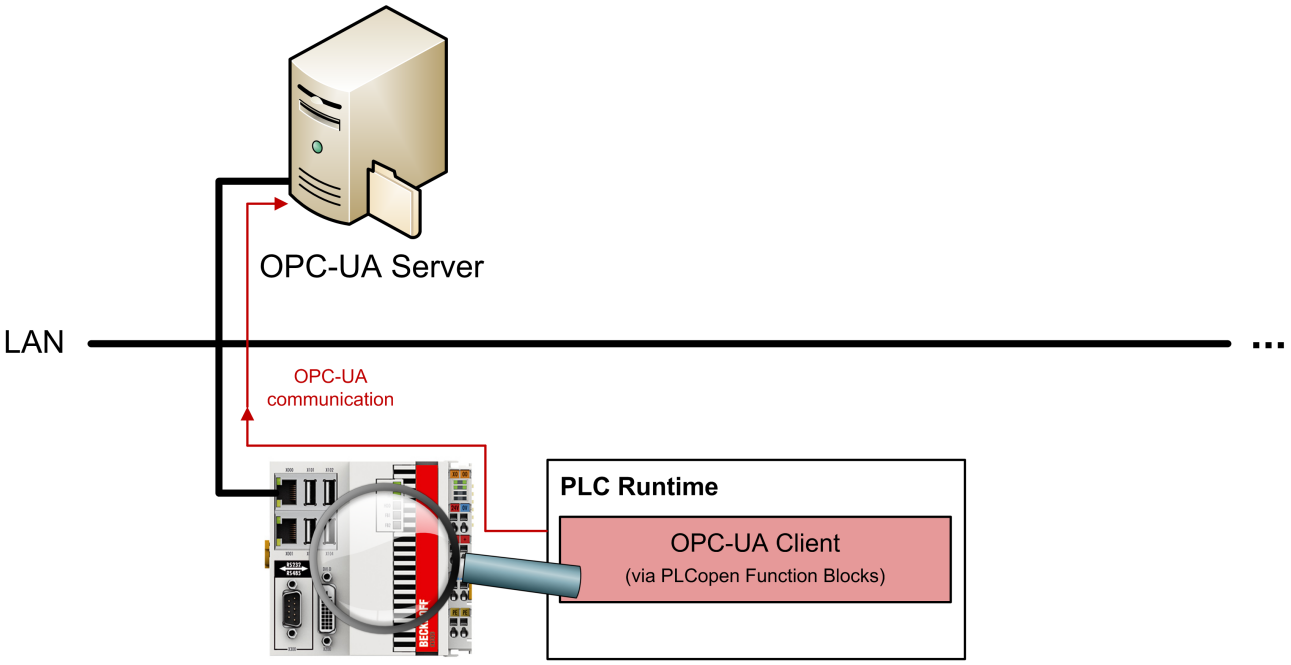
```
<ServerConfigNamespace Exposed="True" MinUserlevel="14" />
<Users>
  <User Level="15">Administrator</User>
  <User Level="10">domain\RegularUser</User>
</Users>
```

In dem vorstehenden Beispiel ist dem lokalen Benutzernamen „Administrator" der Zugriff auf den Konfiguration des Namensraums gestattet, wohingegen das Benutzerkonto „domain\RegularUser" der Windows-Domäne nicht auf diesen Namensraum zugreifen darf, da die Benutzerebene zu niedrig ist.

4.2 Client

4.2.1 Übersicht

Der TwinCAT OPC-UA Client stellt eine OPC-UA Schnittstelle direkt aus der SPS zur Verfügung. Hierdurch wird dem SPS-Programmierer eine standardisierte Kommunikationsschnittstelle zur Verbindung mit OPC-UA Servern über SPS-Funktionsbausteine zur Verfügung gestellt. Die verwendbaren Funktionsbausteine wurden im Rahmen einer PLCopen-Arbeitsgruppe genormt.



Embedded Controller and/or Industrial-PC
WinCE/WES/WinPro

Der OPC-UA Client wird immer zusammen mit der SPS-Bibliothek Tcx_PLCopen_OpcUa verwendet, welche ebenfalls Bestandteil des Produkt-Setups ist. Diese Bibliothek stellt diverse OPC-UA Funktionalitäten zur Verfügung, welche dann aus einem SPS-Programm verwendet werden können.

Die folgende Tabelle gibt einen Überblick über die angebotenen Funktionalitäten:

Funktionalität	Beschreibung	Relevante Funktionsbausteine
Connect / Disconnect	Herstellen und Trennen von Verbindungen zu OPC-UA Servern.	UA_Connect UA_Disconnect
Polling (Read/Write)	Lesen und Schreiben von Variablen im UA-Namensraum im Polling-Modus. Beinhaltet keine Subscriptions.	UA_Connect UA_Disconnect UA_Read UA_Write UA_GetNamespaceIndex UA_NodeGetHandle UA_NodeReleaseHandle
Method-Call	Ausführen von Methoden im UA-Namensraum.	UA_Connect UA_Disconnect UA_MethodGetHandle UA_MethodReleaseHandle UA_MethodCall
Security	Herstellen einer verschlüsselten Verbindung zu einem OPC-UA Server.	UA_Connect UA_Disconnect

Die Schnittstellen jedes Funktionsbausteins sind im Rahmen dieser Dokumentation ausführlich im Bereich SPS-Bibliotheken dokumentiert.

4.2.2 Best practice

4.2.2.1 Wie Kommunikationsparameter zu bestimmen sind

Ganz gleich, welche OPC-UA-Funktionalität Sie implementieren möchten, es gibt doch ein paar Dinge, die bei der SPS-Projektentwicklung hilfreich sind. Es ist allgemeine Praxis, einen grafischen OPC-UA-Client zu verwenden, um bei der Bestimmung der Attribute eines Knotens oder einer Methode zu helfen, die zusammen mit den SPS-Funktionsbausteinen verwendet werden müssen, z. B.:

- Node Identifier
- Node Namespace Index und entsprechender Namespace URI
- Node Data Type
- Method Node ID und Object Node ID

Die folgende Dokumentation verwendet den generischen OPC-UA Client UA-Expert als Beispiel. Dieser Client kann von den Websites von Unified Automation erworben werden unter www.unified-automation.com.

Node Identifier

Eine sehr allgemeine Aufgabe besteht aus dem Lesen oder Schreiben von Variablen – die im Allgemeinen im Hinblick auf OPC-UA als Knoten bezeichnet werden. Knoten sind durch die folgenden drei Elemente gekennzeichnet:

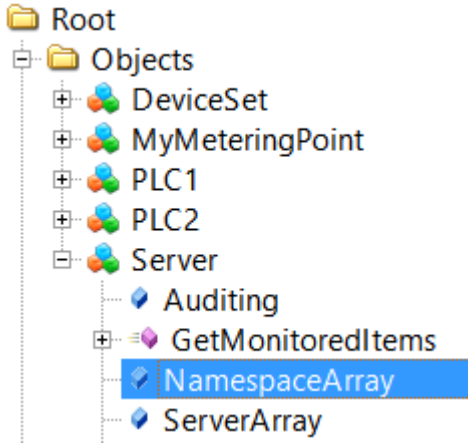
- Namespace Index: Der Namensraum, in dem sich der Knoten befindet, z. B. die SPS-Laufzeit
- Identifier: Eindeutiger Bezeichner des Knotens innerhalb seines Namensraums
- Identifier Type: Es gibt drei verschiedene Typen: String, Guid und Numeric.

Diese Attribute stellen die sogenannte NodeID dar – die Darstellung eines Knotens auf einem OPC-UA Server – und sie werden von manchen Funktionsbausteinen benötigt. Durch Verwendung von UA-Expert können Entwickler diese Attribute einfach durch Verbindung zum OPC-UA Server und Browsen zum gewünschten Knoten bestimmen. Die Attribute sind dann im Attributes-Panel sichtbar, z. B.:

NodeId	NodeId
NamespaceIndex	5
IdentifierType	String
Identifier	MAIN.nCounter

Node Namespace Index und entsprechender Namespace URI

Nach der OPC-UA Spezifikation kann der Namespace Index (wie vorstehend wiedergegeben) ein dynamisch generierter Wert sein. Daher müssen OPC-UA Clients immer den entsprechenden Namespace URI zur Auflösung des Namespace Index verwenden, bevor ein Knotenhandle erfasst wird. Dies erfolgt durch Verwendung des Funktionsbausteins UA_GetNamespaceIndex [▶ 118]. Um diesen Funktionsbaustein jedoch verwenden zu können, muss man den Namespace URI kennen. Der Namespace URI kann einfach durch Verwendung von UA-Expert bestimmt werden. An den OPC-UA Server anschließen und zu folgendem Knoten browsen:



Dieser Knoten enthält Informationen über alle eingetragenen Namespaces auf dem OPC-UA Server. Sie können die entsprechenden Namespace URIs in dem Attributes-Panel sehen.

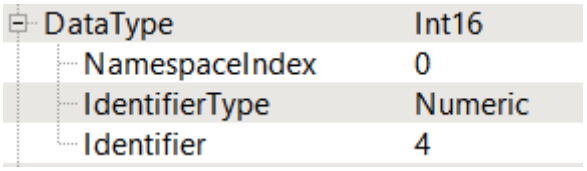
Value	
SourceTimestamp	16.02.2015 08:56:06.350
ServerTimestamp	16.02.2015 09:31:01.945
SourcePicoseconds	0
ServerPicoseconds	0
Value	String Array[9]
[0]	http://opcfoundation.org/UA/
[1]	urn:SvenG-NB04:BeckhoffAutomation:TcOpcUaServer:1
[2]	http://opcfoundation.org/UA/DI/
[3]	http://PLCopen.org/OpcUa/IEC61131-3/
[4]	urn://SVENG-NB04/BeckhoffAutomation/Ua/Typesystem
[5]	urn://SVENG-NB04/BeckhoffAutomation/Ua/PLC1
[6]	urn://SVENG-NB04/BeckhoffAutomation/Ua/PLC2
[7]	http://www.opcfoundation.org/Energy/DataAcquisition/
[8]	http://Beckhoff.com/TwinCAT/TF6100/Server/Configuration

Beispiel: In dem Kapitel über Node Identifier (siehe vorstehend) haben wir eine NodeID gesehen, in der der Namespace Index 5 war. Nach dem vorstehenden Namespace Array ist der entsprechende Namespace URI urn://SVENG-NB04/BeckhoffAutomation/Ua/PLC1. Dieser URI kann nun für den Funktionsbaustein UA_GetNamespaceIndex verwendet werden. Der OPC-UA Server stellt sicher, dass dieser URI immer derselbe bleibt, auch nach einem Neustart. Der vorgenannte Namespace Index kann sich jedoch ändern,

weshalb der Namespace URI und der Funktionsbaustein UA_GetNamespaceIndex immer zur Auflösung des korrekten Namespace Index zur späteren Nutzung mit anderen Funktionsbausteinen genutzt werden sollte, wie z. B. UA_Read [▶ 124], UA_Write [▶ 125].

Node Data Type

Der Datentyp eines Knotens ist erforderlich, um zu sehen, welcher SPS-Datentyp verwendet werden muss, um einen ausgelesenen Wert zuzuordnen oder welcher SPS-Datentyp verwendet werden muss, wenn in einen Knoten geschrieben werden soll. Bei Verwendung von UA-Expert, einfach mit dem OPC-UA Server verbinden, zur Variablen browsen und deren Data Type auf dem Attributes-Panel auslesen, z. B.:



In diesem Fall ist der Data Type Int16, was einem äquivalenten Datentyp in der SPS zugeordnet werden sollte, z. B. INT. Die PLCopen IEC61131-zu-OPC-UA Spezifikation beschreibt das definierte Datentyp-Mapping. Die folgende Tabelle ist ein Auszug aus dieser Spezifikation:

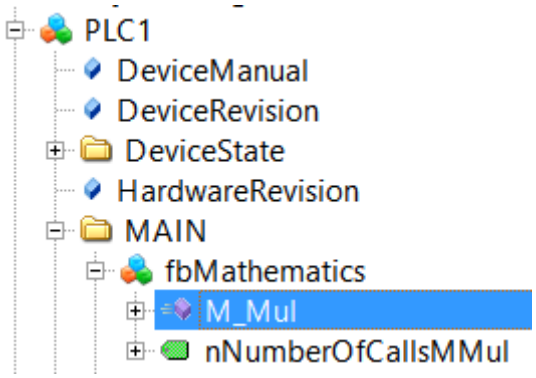
IEC61131 elementare Datentypen	OPC-UA eingebaute Datentypen
BOOL	Boolean
SINT	SByte
USINT	Byte
INT	Int16
UINT	UInt16
DINT	Int32
UDINT	UInt32
LINT	Int64
ULINT	UInt64
BYTE	Byte
WORD	UInt16
DWORD	UInt32
LWORD	UInt64
REAL	Float
LREAL	Double
STRING	String
CHAR	Byte
WSTRING	String
WCHAR	UInt16
DT DATE_AND_TIME	DateTime
DATE	DateTime
TOD TIME_OF_DAY	DateTime
TIME	Double

Method Node ID und Object Node ID

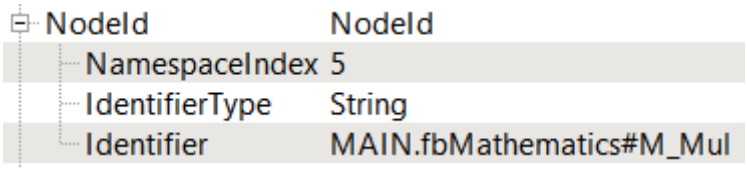
Beim Aufruf von Methoden aus dem OPC-UA Namensraum sind zwei Identifier erforderlich, wenn der Methodenhandle unter Verwendung des Funktionsbausteins UA_MethodGetHandle [▶ 120] erfasst wird:

- Object Node ID: Identifiziert das UA-Objekt, das die Methode enthält
- Method Node ID: Identifiziert die Methode selbst

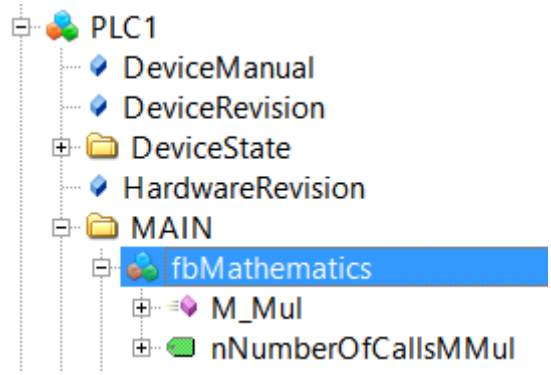
Beide NodeIDs können einfach durch Verwendung von UA-Expert bestimmt werden. An den OPC-UA Server anschließen und zu der gewünschten Methode browsen, z.B.:



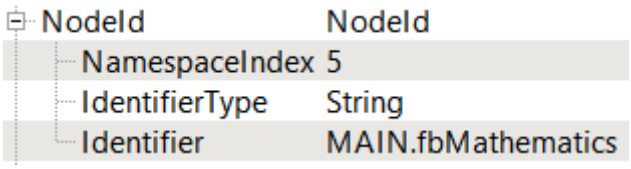
Der Method Identifier kann dann vom Attributes-Panel aus ausgelesen werden:



Browsen Sie anschließend zu dem UA-Objekt, dass diese Methode enthält. In dem vorstehenden Beispiel wäre dies das Objekt fbMathematics:



Der Object Identifier kann dann vom Attributes-Panel aus ausgelesen werden:



4.2.2.2 Wie eine Verbindung hergestellt wird

Der folgende Artikel beschreibt, wie die TcX_PLCopen_OpcUa Funktionsbausteine verwendet werden sollten, um eine Verbindung zu einem lokalen oder entfernten OPC-UA Server herzustellen. Diese Verbindung kann dann verwendet werden, um weitere Funktionalitäten aufzurufen, z. B. Knoten auslesen oder schreiben, oder Methoden aufrufen. Der Artikel beinhaltet folgende Themen:

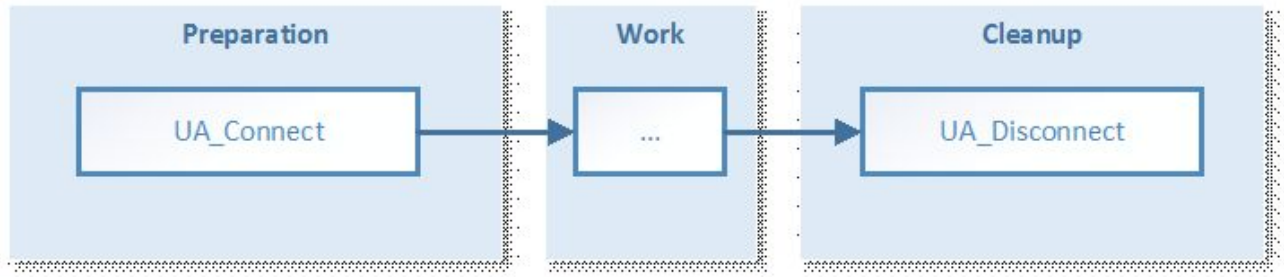
- Übersicht
- Schematischer Arbeitsablauf
- Zu berücksichtigende Punkte
- Code-Ausschnitt

Übersicht

Die folgenden Funktionsbausteine sind erforderlich, um eine Verbindung zu einem OPC-UA Server herzustellen und später die Sitzung zu unterbrechen: UA_Connect [▶ 116], UA_Disconnect [▶ 117]. Wir empfehlen dringend, zunächst unseren Artikel Wie Kommunikationsparameter zu bestimmen sind [▶ 87] durchzulesen, um in der Lage zu sein, bestimmte UA-Funktionalitäten zu verstehen, z. B. wie man Node Identifier bestimmen kann.

Schematischer Arbeitsablauf

Der schematische Arbeitsablauf jedes TwinCAT OPC-UA Client kann in drei verschiedene Phasen kategorisiert werden: Preparation, Work und Cleanup. Der in diesem Artikel beschriebene Verwendungsfall kann wie folgt visualisiert werden:



Zu berücksichtigende Punkte

Der Funktionsbaustein UA_Connect erfordert die folgenden Informationen, um in der Lage zu sein, eine Verbindung zu einem lokalen oder entfernten OPC-UA Server herzustellen:

- Server URL
- Session Connect Information

Die Server URL besteht grundsätzlich aus einem Präfix, einem Hostnamen und einem Port. Das Präfix beschreibt das OPC-UA Transportprotokoll, das für die Verbindung verwendet werden sollte, z. B. “opc.tcp://” für eine binäre TCP-Verbindung (Standard). Der Hostname bzw. IP-Adressenteil beschreibt die Adressinformationen des OPC-UA Zielservers, z. B. “192.168.1.1” oder “CX-12345”. Die Portnummer ist der Zielport des OPC-UA Servers, z. B. “4840“. Insgesamt kann die Server URL wie folgt aussehen: opc.tcp://CX-12345:4840.

Code-Ausschnitt

Deklaration:

```
(* Declarations for UA_Connect *)
fbUA_Connect : UA_Connect;
SessionConnectInfo : ST_UASessionConnectInfo;
nConnectionHdl : DWORD;

(* Declarations for UA_Disconnect *)
fbUA_Disconnect : UA_Disconnect;

(* Declarations for state machine and output handling *)
iState : INT;
bDone: BOOL;
bBusy: BOOL;
bError : BOOL;
nErrorID : DWORD;
```

Implementierung:

```
CASE iState OF

  0:
      bError := FALSE;
      nErrorID := 0;
      SessionConnectInfo.tConnectTimeout := T#1M;
      SessionConnectInfo.tSessionTimeout := T#1M;
      SessionConnectInfo.sApplicationName := '';
      SessionConnectInfo.sApplicationUri := '';
```

```
      SessionConnectInfo.eSecurityMode := eUASecurityMsgMode_None;
      SessionConnectInfo.eSecurityPolicyUri := eUASecurityPolicy_None;
      SessionConnectInfo.eTransportProfileUri := eUATransportProfileUri_UATcp;
      stNodeAddInfo.nIndexRangeCount := nIndexRangeCount;
      stNodeAddInfo.stIndexRange := stIndexRange;
      iState := iState + 1;

  1:
    fbUA_Connect(
      Execute := TRUE,
      ServerURL := 'opc.tcp://192.168.1.1:4840',
      SessionConnectInfo := SessionConnectInfo,
      Timeout := T#5S,
      ConnectionHdl => nConnectionHdl);
    IF NOT fbUA_Connect.Busy THEN
      fbUA_Connect(Execute := FALSE);
      IF NOT fbUA_Connect.Error THEN
        iState := iState + 1;
      ELSE
        bError := TRUE;
        nErrorID := fbUA_Connect.ErrorID;
        nConnectionHdl := 0;
        iState := 0;
      END_IF
    END_IF

  2:
    fbUA_Disconnect(
      Execute := TRUE,
      ConnectionHdl:= nConnectionHdl);

    IF NOT fbUA_Disconnect.Busy THEN
      fbUA_Disconnect(Execute := FALSE);
      IF NOT fbUA_Disconnect.Error THEN
        iState := 0;
      ELSE
        bError := TRUE;
        nErrorID := fbUA_Disconnect.ErrorID;
        iState := 0;
        nConnectionHdl := 0;
      END_IF
    END_IF

END_CASE
```

4.2.2.3 Wie die Knoten auszulesen sind

Der folgende Artikel beschreibt, wie die TcX_PLCopen_OpcUa Funktionsbausteine verwendet werden sollten, um einen OPC-UA Knoten von einem lokalen oder entfernten OPC-UA Server auszulesen. Wir empfehlen, zunächst unseren Artikel Wie eine Verbindung hergestellt wird [▶ 90] durchzulesen. Der Artikel beinhaltet folgende Themen:

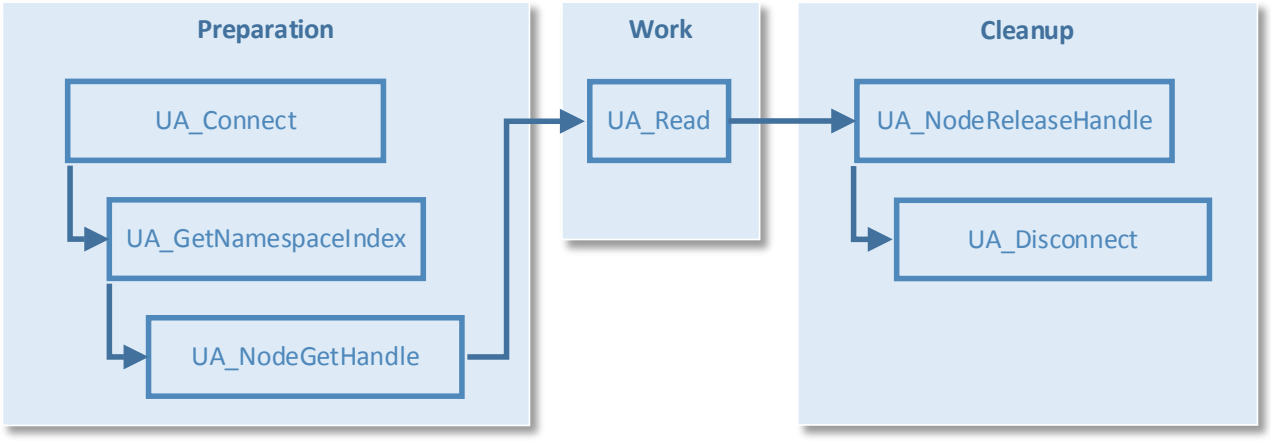
- Übersicht
- Schematischer Arbeitsablauf
- Zu berücksichtigende Punkte
- Code-Ausschnitt

Übersicht

Die folgenden Funktionsbausteine sind erforderlich, um eine Verbindung zu einem OPC-UA Server herzustellen, UA-Knoten auszulesen und später die Sitzung zu unterbrechen: UA_Connect [▶ 116], UA_GetNamespaceIndex [▶ 118], UA_NodeGetHandle [▶ 122], UA_Read [▶ 124], UA_NodeReleaseHandle [▶ 123], UA_Disconnect [▶ 117]. Wir empfehlen dringend, zunächst unseren Artikel Wie Kommunikationsparameter zu bestimmen sind [▶ 87] durchzulesen, um in der Lage zu sein, bestimmte UA-Funktionalitäten zu verstehen, z. B. wie man Node Identifier bestimmen kann.

Schematischer Arbeitsablauf

Der schematische Arbeitsablauf jedes TwinCAT OPC-UA Client kann in drei verschiedene Phasen kategorisiert werden: Preparation, Work und Cleanup. Der in diesem Artikel beschriebene Verwendungsfall kann wie folgt visualisiert werden:



Zu berücksichtigende Punkte

Der Funktionsbaustein UA_Connect erfordert die folgenden Informationen, um in der Lage zu sein, eine Verbindung zu einem lokalen oder entfernten OPC-UA Server herzustellen – was bereits in dem Artikel Wie eine Verbindung hergestellt wird [▶ 90] behandelt wurde.

- Server URL
- Session Connect Information

Der Funktionsbaustein UA_GetNamespaceIndex erfordert einen Connection Handle (von UA_Connect) und Namespace URI zur Auflösung in einen Namespace Index, was später von UA_NodeGetHandle verwendet wird, um einen Knotenhandle zu erfassen. Das Konzept von Namespace URIs und Namespace Index wurde bereits in dem Artikel Wie Kommunikationsparameter zu bestimmen sind [▶ 87] behandelt.

Der Funktionsbaustein UA_NodeGetHandle erfordert einen Connection Handle (von UA_Connect) und NodeID (von ST_UANodeID), um einen Knotenhandle zu erfassen. Die Konzepte hinter einem NodeID wurden bereits in dem Artikel Wie Kommunikationsparameter zu bestimmen sind [▶ 87] behandelt.

Der Funktionsbaustein UA_Read erfordert einen Verbindungshandle (von UA_Connect), Knotenhandle (von UA_NodeGetHandle) und Zeiger zur Zielvariablen (wo der ausgelesene Wert gespeichert werden sollte). Stellen Sie dabei sicher, dass die Zielvariable den korrekten Datentyp aufweist, wie dies in dem Artikel Wie Kommunikationsparameter zu bestimmen sind [▶ 87] beschrieben ist.

Der Funktionsbaustein UA_NodeReleaseHandle erfordert den Verbindungshandle (von UA_Connect) und Knotenhandle (von UA_NodeGetHandle).

Code-Ausschnitt

Deklaration:

```
(* Declarations for UA_GetNamespaceIndex *)
fbUA_GetNamespaceIndex : UA_GetNamespaceIndex;
nNamespaceIndex : UINT;

(* Declarations for UA_NodeGetHandle *)
fbUA_NodeGetHandle : UA_NodeGetHandle;
NodeID : ST_UANodeID;
nNodeHdl : DWORD;

(* Declarations for UA_Read *)
fbUA_Read : UA_Read;
stIndexRange : ARRAY [1..nMaxIndexRange] OF ST_UAIndexRange;
nIndexRangeCount : UINT;
stNodeAddInfo : ST_UANodeAdditionalInfo;
sNodeIdentifier : STRING(MAX_STRING_LENGTH) := 'MAIN.nCounter';
nReadData : INT;
```

```
cbDataRead : UDINT;

(* Declarations for UA_NodeReleaseHandle *)
fbUA_NodeReleaseHandle : UA_NodeReleaseHandle;
```

Implementierung:

```
CASE iState OF
  0:
    [...]

  2: (* GetNS Index *)
    fbUA_GetNamespaceIndex(
      Execute := TRUE,
      ConnectionHdl := nConnectionHdl,
      NamespaceUri := sNamespaceUri,
      NamespaceIndex => nNamespaceIndex
      );
    IF NOT fbUA_GetNamespaceIndex.Busy THEN
      fbUA_GetNamespaceIndex(Execute := FALSE);
      IF NOT fbUA_GetNamespaceIndex.Error THEN
        iState := iState + 1;
      ELSE
        bError := TRUE;
        nErrorID := fbUA_GetNamespaceIndex.ErrorID;
        iState := 6;
      END_IF
    END_IF

  3: (* UA_NodeGetHandle *)
    NodeID.eIdentifierType := eUAIdentifierType_String;
    NodeID.nNamespaceIndex := nNamespaceIndex;
    NodeID.sIdentifier := sNodeIdentifier;
    fbUA_NodeGetHandle(
      Execute := TRUE,
      ConnectionHdl := nConnectionHdl,
      NodeID := NodeID,
      NodeHdl => nNodeHdl);
    IF NOT fbUA_NodeGetHandle.Busy THEN
      fbUA_NodeGetHandle(Execute := FALSE);
      IF NOT fbUA_NodeGetHandle.Error THEN
        iState := iState + 1;
      ELSE
        bError := TRUE;
        nErrorID := fbUA_NodeGetHandle.ErrorID;
        iState := 6;
      END_IF
    END_IF

  4: (* UA_Read *)
    fbUA_Read(
      Execute := TRUE,
      ConnectionHdl := nConnectionHdl,
      NodeHdl := nNodeHdl,
      cbData := SIZEOF(nReadData),
      stNodeAddInfo := stNodeAddInfo,
      pVariable := ADR(nReadData));
    IF NOT fbUA_Read.Busy THEN
      fbUA_Read( Execute := FALSE, cbData_R => cbDataRead);
      IF NOT fbUA_Read.Error THEN
        iState := iState + 1;
      ELSE
        bError := TRUE;
        nErrorID := fbUA_Read.ErrorID;
        iState := 6;
      END_IF
    END_IF

  5: (* Release Node Handle *)
    fbUA_NodeReleaseHandle(
      Execute := TRUE,
      ConnectionHdl := nConnectionHdl,
      NodeHdl := nNodeHdl);
    IF NOT fbUA_NodeReleaseHandle.Busy THEN
      fbUA_NodeReleaseHandle(Execute := FALSE);
      IF NOT fbUA_NodeReleaseHandle.Error THEN
        iState := iState + 1;
      ELSE
        bError := TRUE;
        nErrorID := fbUA_NodeReleaseHandle.ErrorID;
```

```
        iState := 6;
      END_IF
    END_IF

  6:
    [...]

END_CASE
```

4.2.2.4 Wie Knoten zu schreiben sind

Der folgende Artikel beschreibt, wie die TcX_PLCopen_OpcUa Funktionsbausteine verwendet werden sollten, um Werte in einem OPC-UA Knoten von einem lokalen oder entfernten OPC-UA Server zu schreiben. Wir empfehlen, zunächst unseren Artikel Wie eine Verbindung hergestellt wird [▶ 90] durchzulesen. Der Artikel beinhaltet folgende Themen:

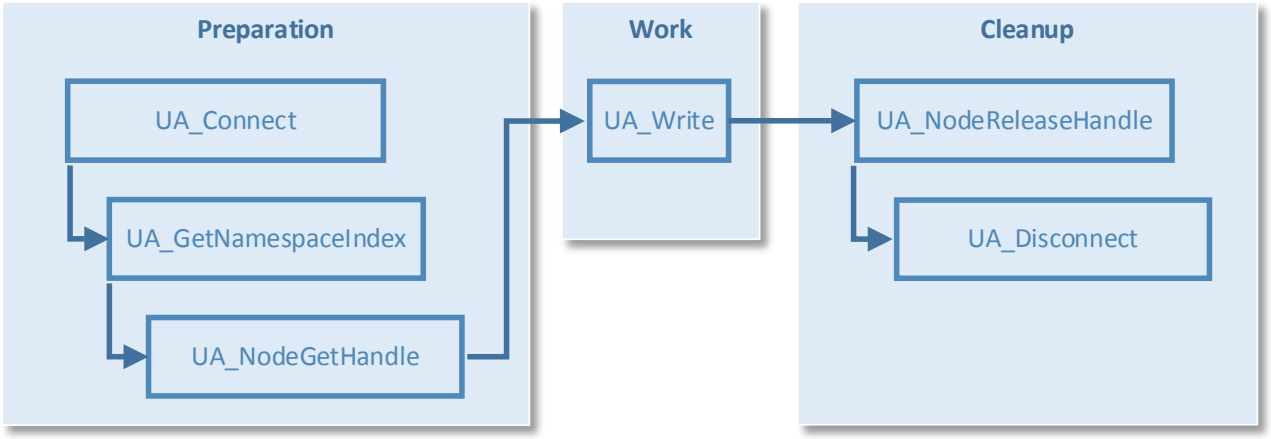
- Übersicht
- Schematischer Arbeitsablauf
- Zu berücksichtigende Punkte
- Code-Ausschnitt

Übersicht

Die folgenden Funktionsbausteine sind erforderlich, um eine Verbindung zu einem OPC-UA Server herzustellen, UA-Knoten zu schreiben und später die Sitzung zu unterbrechen: UA_Connect [▶ 116], UA_GetNamespaceIndex [▶ 118], UA_NodeGetHandle [▶ 122], UA_Write [▶ 125], UA_NodeReleaseHandle [▶ 123], UA_Disconnect [▶ 117]. Wir empfehlen dringend, zunächst unseren Artikel Wie Kommunikationsparameter zu bestimmen sind [▶ 87] durchzulesen, um in der Lage zu sein, bestimmte UA-Funktionalitäten zu verstehen, z. B. wie man Node Identifier bestimmen kann.

Schematischer Arbeitsablauf

Der schematische Arbeitsablauf jedes TwinCAT OPC-UA Client kann in drei verschiedene Phasen kategorisiert werden: Preparation, Work und Cleanup. Der in diesem Artikel beschriebene Verwendungsfall kann wie folgt visualisiert werden:



Zu berücksichtigende Punkte

Der Funktionsbaustein UA_Connect erfordert die folgenden Informationen, um in der Lage zu sein, eine Verbindung zu einem lokalen oder entfernten OPC-UA Server herzustellen – was bereits in dem Artikel Wie eine Verbindung hergestellt wird [▶ 90] behandelt wurde.

- Server URL
- Session Connect Information

Der Funktionsbaustein UA_GetNamespaceIndex erfordert einen Connection Handle (von UA_Connect) und Namespace URI zur Auflösung in einen Namespace Index, was später von UA_NodeGetHandle verwendet wird, um einen Knotenhandle zu erfassen. Das Konzept von Namespace URIs und Namespace Index wurde bereits in dem Artikel Wie Kommunikationsparameter zu bestimmen sind [▶ 87] behandelt.

Der Funktionsbaustein UA_NodeGetHandle erfordert einen Connection Handle (von UA_Connect) und NodeID (von ST_UANodeID), um einen Knotenhandle zu erfassen. Die Konzepte hinter einem NodeID wurden bereits in dem Artikel Wie Kommunikationsparameter zu bestimmen sind [▶ 87] behandelt.

Der Funktionsbaustein UA_Write erfordert einen Verbindungshandle (von UA_Connect), Knotenhandle (von UA_NodeGetHandle) und Zeiger zu einer Variablen, die den Wert enthält, der geschrieben werden soll. Stellen Sie dabei sicher, dass die Zielvariable den korrekten Datentyp aufweist, wie dies in dem Artikel Wie Kommunikationsparameter zu bestimmen sind [▶ 87] beschrieben ist.

Der Funktionsbaustein UA_NodeReleaseHandle erfordert den Verbindungshandle (von UA_Connect) und Knotenhandle (von UA_NodeGetHandle).

Code-Ausschnitt

Deklaration:

```
(* Declarations for UA_GetNamespaceIndex *)
fbUA_GetNamespaceIndex : UA_GetNamespaceIndex;
nNamespaceIndex : UINT;

(* Declarations for UA_NodeGetHandle *)
fbUA_NodeGetHandle : UA_NodeGetHandle;
NodeID : ST_UANodeID;
nNodeHdl : DWORD;

(* Declarations for UA_Write *)
fbUA_Write : UA_Write;
stIndexRange : ARRAY [1..nMaxIndexRange] OF ST_UAIndexRange;
nIndexRangeCount : UINT;
stNodeAddInfo : ST_UANodeAdditionalInfo;
sNodeIdentifier: STRING(MAX_STRING_LENGTH) := 'MAIN.nNumber';
nWriteData: INT := 42;

(* Declarations for UA_NodeReleaseHandle *)
fbUA_NodeReleaseHandle : UA_NodeReleaseHandle;
```

Implementierung:

```
CASE iState OF
  0:
    [...]

  2: (* GetNS Index *)
    fbUA_GetNamespaceIndex(
      Execute := TRUE,
      ConnectionHdl := nConnectionHdl,
      NamespaceUri := sNamespaceUri,
      NamespaceIndex => nNamespaceIndex
      );
    IF NOT fbUA_GetNamespaceIndex.Busy THEN
      fbUA_GetNamespaceIndex(Execute := FALSE);
      IF NOT fbUA_GetNamespaceIndex.Error THEN
        iState := iState + 1;
      ELSE
        bError := TRUE;
        nErrorID := fbUA_GetNamespaceIndex.ErrorID;
        iState := 6;
      END_IF
    END_IF

  3: (* UA_NodeGetHandle *)
    NodeID.eIdentifierType := eUAIdentifierType_String;
    NodeID.nNamespaceIndex := nNamespaceIndex;
    NodeID.sIdentifier := sNodeIdentifier;
    fbUA_NodeGetHandle(
      Execute := TRUE,
      ConnectionHdl := nConnectionHdl,
      NodeID := NodeID,
      NodeHdl => nNodeHdl);
    IF NOT fbUA_NodeGetHandle.Busy THEN
```

```
      fbUA_NodeGetHandle(Execute := FALSE);
      IF NOT fbUA_NodeGetHandle.Error THEN
        iState := iState + 1;
      ELSE
        bError := TRUE;
        nErrorID := fbUA_NodeGetHandle.ErrorID;
        iState := 6;
      END_IF
    END_IF

  4: (* UA_Write *)
    fbUA_Write(
      Execute := TRUE,
      ConnectionHdl := nConnectionHdl,
      NodeHdl := nNodeHdl,
      stNodeAddInfo := stNodeAddInfo,
      cbData := SIZEOF(nWriteData),
      pVariable := ADR(nWriteData));
    IF NOT fbUA_Write.Busy THEN
      fbUA_Write(
        Execute := FALSE,
        pVariable:= ADR(nWriteData));
      IF NOT fbUA_Write.Error THEN
        iState := iState + 1;
      ELSE
        bError := TRUE;
        nErrorID := fbUA_Write.ErrorID;
        iState := 6;
      END_IF
    END_IF

  5: (* Release Node Handle *)
    fbUA_NodeReleaseHandle(
      Execute := TRUE,
      ConnectionHdl := nConnectionHdl,
      NodeHdl := nNodeHdl);
    IF NOT fbUA_NodeReleaseHandle.Busy THEN
      fbUA_NodeReleaseHandle(Execute := FALSE);
      IF NOT fbUA_NodeReleaseHandle.Error THEN
        iState := iState + 1;
      ELSE
        bError := TRUE;
        nErrorID := fbUA_NodeReleaseHandle.ErrorID;
        iState := 6;
      END_IF
    END_IF

  6:
    [...]

END_CASE
```

4.2.2.5 Wie Methoden aufzurufen sind

Der folgende Artikel beschreibt, wie die TcX_PLCopen_OpcUa Funktionsbausteine verwendet werden sollten, um Methoden auf einem lokalen oder entfernten OPC-UA Server aufzurufen. Wir empfehlen, zunächst unseren Artikel Wie eine Verbindung hergestellt wird [▶ 90] durchzulesen. Der Artikel beinhaltet folgende Themen:

- Übersicht
- Schematischer Arbeitsablauf
- Zu berücksichtigende Punkte
- Code-Ausschnitt

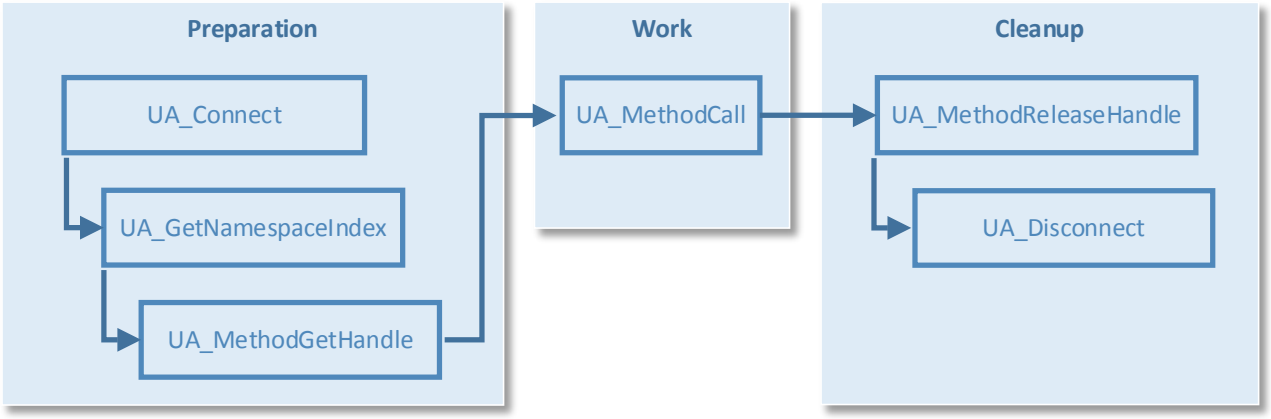
Übersicht

Die folgenden Funktionsbausteine sind erforderlich, um eine Verbindung zu einem OPC-UA Server herzustellen, UA-Methoden aufzurufen und später die Sitzung zu unterbrechen: UA_Connect [▶ 116], UA_GetNamespaceIndex [▶ 118], UA_MethodGetHandle [▶ 120], UA_MethodCall [▶ 119],

UA_MethodReleaseHandle [▶ 121], UA_Disconnect [▶ 117]. Wir empfehlen dringend, zunächst unseren Artikel Wie Kommunikationsparameter zu bestimmen sind [▶ 87] durchzulesen, um in der Lage zu sein, bestimmte UA-Funktionalitäten zu verstehen, z. B. wie man Method Identifier bestimmen kann.

Schematischer Arbeitsablauf

Der schematische Arbeitsablauf jedes TwinCAT OPC-UA Client kann in drei verschiedene Phasen kategorisiert werden: Preparation, Work und Cleanup. Der in diesem Artikel beschriebene Verwendungsfall kann wie folgt visualisiert werden:



Zu berücksichtigende Punkte

Der Funktionsbaustein UA_Connect erfordert die folgenden Informationen, um in der Lage zu sein, eine Verbindung zu einem lokalen oder entfernten OPC-UA Server herzustellen – was bereits in dem Artikel Wie eine Verbindung hergestellt wird [▶ 90] behandelt wurde.

- Server URL
- Session Connect Information

Der Funktionsbaustein UA_GetNamespaceIndex erfordert einen Connection Handle (von UA_Connect) und Namespace URI zur Auflösung in einen Namespace Index, was später von UA_NodeGetHandle verwendet wird, um einen Knotenhandle zu erfassen. Das Konzept von Namespace URIs und Namespace Index wurde bereits in dem Artikel Wie Kommunikationsparameter zu bestimmen sind [▶ 87] behandelt.

Der Funktionsbaustein UA_MethodGetHandle erfordert einen Connection Handle (von UA_Connect), eine Object NodeID und Method NodeID, um einen Methodenhandle zu erfassen. Die Konzepte hinter einer Object und Method NodeID wurden bereits in dem Artikel Wie Kommunikationsparameter zu bestimmen sind [▶ 87] behandelt.

Der Funktionsbaustein UA_MethodCall erfordert einen Verbindungshandle (von UA_Connect), Methodenhandle (von UA_MethodGetHandle) und Informationen über die Eingangs- und Ausgangsargumente der Methode, die aufgerufen werden soll. Informationen über die Eingangsargumente werden durch die pInputArgInfo und pInputArgData Eingangsparameter von UA_MethodCall repräsentiert. Information über die Ausgangsparameter werden durch die pOutputArgInfo und pOutputArgData Eingangsparameter von UA_MethodCall repräsentiert. Der Eingangsparameter pOutputArgInfoAndData stellt dann einen Zeiger zu einer Struktur dar, die die Ergebnisse des Methodenaufrufs enthält, einschließlich aller Ausgangsparameter. In dem nachstehenden Code-Ausschnitt werden die pInputArgInfo und pInputArgData Parameter in der M_Init() Methode berechnet und erstellt.

Der Funktionsbaustein UA_NodeReleaseHandle erfordert den Verbindungshandle (von UA_Connect) und Methodenhandle (von UA_MethodGetHandle).

Code-Ausschnitt

M_Init() – Initialisierungsmethode des Funktionsbausteins, der den UA Methodenaufruf enthält

```
MEMSET(ADR(nInputData),0,SIZEOF(nInputData));
nArg := 1;
```

```
(********** Input parameter 1 **********)
InputArguments[nArg].DataType := eUAType_Int16;
InputArguments[nArg].ValueRank := -1; (* Scalar = -1 or Array *)
InputArguments[nArg].ArrayDimensions[1] := 0; (* Number of Dimension in case its an array *)
InputArguments[nArg].nLenData := SIZEOF(numberIn1); (* Length if its a STRING *)
IF nOffset + SIZEOF(numberIn1) > nInputArgSize THEN
  bInputDataError := TRUE;
  RETURN;
ELSE
  MEMCPY(ADR(nInputData)+nOffset,ADR(numberIn1),SIZEOF(numberIn1)); (* VALUE in BYTES FORM *)
  nOffset := nOffset + SIZEOF(numberIn1);
END_IF
nArg := nArg + 1;

(********** Input parameter 2 **********)
InputArguments[nArg].DataType := eUAType_Int16;
InputArguments[nArg].ValueRank := -1; (* Scalar = -1 or Array *)
InputArguments[nArg].ArrayDimensions[1] := 0; (* Number of Dimension in case its an array *)
InputArguments[nArg].nLenData := SIZEOF(numberIn2); (* Length if its a STRING *)
IF nOffset + SIZEOF(numberIn2) > nInputArgSize THEN
  bInputDataError := TRUE;
  RETURN;
ELSE
  MEMCPY(ADR(nInputData)+nOffset,ADR(numberIn2),SIZEOF(numberIn2));(* VALUE in BYTES FORM *)
  nOffset := nOffset + SIZEOF(numberIn2);
END_IF

cbWriteData := nOffset;
```

Deklaration:

```
(* Declarations for UA_GetNamespaceIndex *)
fbUA_GetNamespaceIndex : UA_GetNamespaceIndex;
nNamespaceIndex : UINT;

(* Declarations for UA_MethodGetHandle *)
fbUA_MethodGetHandle: UA_MethodGetHandle;
ObjectNodeID: ST_UANodeID;
MethodNodeID: ST_UANodeID;
nMethodHdl: DWORD;

(* Declarations for UA_MethodCall *)
fbUA_MethodCall: UA_MethodCall;
sObjectNodeIdIdentifier : STRING(MAX_STRING_LENGTH) := 'MAIN.fbMathematics';
sMethodNodeIdIdentifier : STRING(MAX_STRING_LENGTH) := 'MAIN.fbMathematics#M_Mul';
nAdrWriteData: PVOID;
numberIn1: INT := 42; // change according to input value and data type
numberIn2: INT := 42; // change according to input value and data type
numberOutPro: DINT; // result (output parameter of M_Mul())
cbWriteData: UDINT; // calculated automatically by M_Init()
InputArguments: ARRAY[1..2] OF ST_UAMethodArgInfo; // change according to input parameters
stOutputArgInfo: ARRAY[1..1] OF ST_UAMethodArgInfo; // change according to output parameters
stOutputArgInfoAndData: ST_OutputArgInfoAndData;
nInputData: ARRAY[1..4] OF BYTE; // numberIn1(INT16)(2) + numberIn2(INT16)(2)
nOffset: UDINT; // calculated by M_Init()
nArg: INT; // used by M_Init()

(* Declarations for UA_MethodReleaseHandle *)
fbUA_MethodReleaseHandle: UA_MethodReleaseHandle;
```

Implementierung:

```
CASE iState OF
  0:
    [...]

  2: (* GetNS Index *)
    fbUA_GetNamespaceIndex(
      Execute := TRUE,
      ConnectionHdl := nConnectionHdl,
      NamespaceUri := sNamespaceUri,
      NamespaceIndex => nNamespaceIndex);
    IF NOT fbUA_GetNamespaceIndex.Busy THEN
      fbUA_GetNamespaceIndex(Execute := FALSE);
      IF NOT fbUA_GetNamespaceIndex.Error THEN
        iState := iState + 1;
      ELSE
        bError := TRUE;
        nErrorID := fbUA_GetNamespaceIndex.ErrorID;
        iState := 7;
```

```
    END_IF
  END_IF

3: (* Get Method Handle *)
  ObjectNodeID.eIdentifierType := eUAIdentifierType_String;
  ObjectNodeID.nNamespaceIndex := nNamespaceIndex;
  ObjectNodeID.sIdentifier := sObjectNodeIdIdentifier;
  MethodNodeID.eIdentifierType := eUAIdentifierType_String;
  MethodNodeID.nNamespaceIndex := nNamespaceIndex;
  MethodNodeID.sIdentifier := sMethodNodeIdIdentifier;

  M_Init();

  IF bInputDataError = FALSE THEN
    iState := iState + 1;
  ELSE
    bBusy := FALSE;
    bError := TRUE;
    nErrorID := 16#70A; //out of memory
  END_IF

4: (* Method Get Handle *)
  fbUA_MethodGetHandle(
    Execute := TRUE,
    ConnectionHdl := nConnectionHdl,
    ObjectNodeID := ObjectNodeID,
    MethodNodeID := MethodNodeID,
    MethodHdl => nMethodHdl);
  IF NOT fbUA_MethodGetHandle.Busy THEN
    fbUA_MethodGetHandle(Execute := FALSE);
    IF NOT fbUA_MethodGetHandle.Error THEN
      iState := iState + 1;
    ELSE
      bError := TRUE;
      nErrorID := fbUA_MethodGetHandle.ErrorID;
      iState := 6;
    END_IF
  END_IF

5: (* Method Call *)
  stOutputArgInfo[1].nLenData := SIZEOF(stOutputArgInfoAndData.pro);
  fbUA_MethodCall(
    Execute := TRUE,
    ConnectionHdl := nConnectionHdl,
    MethodHdl := nMethodHdl,
    nNumberOfInputArguments := nNumberOfInputArguments,
    pInputArgInfo := ADR(InputArguments),
    cbInputArgInfo := SIZEOF(InputArguments),
    pInputArgData := ADR(nInputData),
    cbInputArgData := cbWriteData,
    pInputWriteData := 0,
    cbInputWriteData := 0,
    nNumberOfOutputArguments := nNumberOfOutputArguments,
    pOutputArgInfo := ADR(stOutputArgInfo),
    cbOutputArgInfo := SIZEOF(stOutputArgInfo),
    pOutputArgInfoAndData := ADR(stOutputArgInfoAndData),
    cbOutputArgInfoAndData := SIZEOF(stOutputArgInfoAndData));
  IF NOT fbUA_MethodCall.Busy THEN
    fbUA_MethodCall(Execute := FALSE);
    IF NOT fbUA_MethodCall.Error THEN
      iState := iState + 1;
      numberOutPro := stOutputArgInfoAndData.pro;
    ELSE
      bError := TRUE;
      nErrorID := fbUA_MethodCall.ErrorID;
      iState := 6;
    END_IF
  END_IF

6: (* Release Method Handle *)
  fbUA_MethodReleaseHandle(
    Execute := TRUE,
    ConnectionHdl := nConnectionHdl,
    MethodHdl := nMethodHdl);
  IF NOT fbUA_MethodReleaseHandle.Busy THEN
    fbUA_MethodReleaseHandle(Execute := FALSE);
    bBusy := FALSE;
    IF NOT fbUA_MethodReleaseHandle.Error THEN
      iState := 7;
    ELSE
```

```
        bError := TRUE;
        nErrorID := fbUA_MethodReleaseHandle.ErrorID;
        iState := 7;
      END_IF
    END_IF

  7:
    [...]

END_CASE
```

4.2.3 Sicherheit

4.2.3.1 Übersicht

Sicherheit war eine zentrale Anforderung bei der Entwicklung von OPC-UA. Sie wird in verschiedenen Bereichen adressiert:

- Verschlüsselung
- Integrität
- Authentifizierung

Die Vertraulichkeit der ausgetauschten Information wird durch die Verschlüsselung der ausgetauschten Nachrichten sichergestellt. Dabei werden moderne Kryptographie-Algorithmen verwendet. Um auch den zukünftigen Sicherheitsanforderungen gewachsen zu sein können im nachhinein noch stärkere und modernere Algorithmen einer Anwendung hinzugefügt werden, ohne das Protokoll zu verändern. Entsprechend der Anforderungen der jeweiligen Anwendung können verschiedene Sicherheitsstufen gewählt werden. In einigen Bereichen ist es ausreichend die Nachrichten zu signieren um Änderungen durch Dritte auszuschließen, in anderen Fällen dürfen Daten auch nicht von Dritten gelesen werden und ein zusätzliches Verschlüsseln der Nachrichten ist erforderlich. TF6100 OPC-UA bietet die folgenden Security-Level (Security-Endpunkte) an, mit welchen Clients sich verbinden können:

- None: Keine Sicherheit
- Sign: Signiert Nachrichten
- Sign&Encrypt: Signiert und verschlüsselt Nachrichten

Das Signieren der Nachrichten verhindert das Ändern des Inhalts einer Nachricht durch einen Dritten. So wird verhindert, daß Beispielsweise eine Schreibanweisung zum Öffnen eines Schalters durch einen Dritten verfälscht wird und der Schalter statt dessen geschlossen wird.

OPC-UA Anwendungen identifizieren sich über so genannte Software und Applikationsinstanz-Zertifikate. Mit Hilfe von Software-Zertifikaten ist es möglich bestimmten Client Anwendungen einen erweiterten Zugriff auf Informationen eines OPC-UA Servers zu geben, beispielsweise für das Engineering eines OPC-UA Servers. Durch Applikationsinstanz-Zertifikate kann sichergestellt werden, daß ein OPC-UA Server nur mit vorkonfigurierten Clients kommuniziert. Ein Client kann durch das Applikationsinstanz-Zertifikat des Servers sicherstellen, daß er mit dem richtigen Server spricht (ähnlich der Zertifikate eines Web-Browsers). Die Berücksichtigung dieser Zertifikate ist optional, d. h. ein OPC-UA Server kann auch jedem Client den gleichen Zugriff abhängig der Benutzerrechte geben.

Der TwinCAT OPC-UA Client und TwinCAT OPC-UA Server erzeugen ein selbst-signiertes Zertifikat während des ersten Startvorgangs. Dieses Zertifikat besteht aus einem privaten und einem öffentlichen Schlüssel. Der private Schlüssel wird in dem Verzeichnis \PKI\CA\private und der entsprechende öffentliche Schlüssel unter \PKI\CA\certs abgelegt. Wenn ein beliebiger OPC-UA Client eine gesicherte Verbindung über einen der Security-Endpunkte (Sign, Sign&Encrypt) mit dem Server herstellen will, muss der Client den öffentlichen Schlüssel des OPC-UA Servers kennen. Umgekehrt muss der OPC-UA Server den öffentlichen Schlüssel des Clients kennen. Dieser sogenannte Schlüsselaustausch wird in den folgenden Artikeln beschrieben.

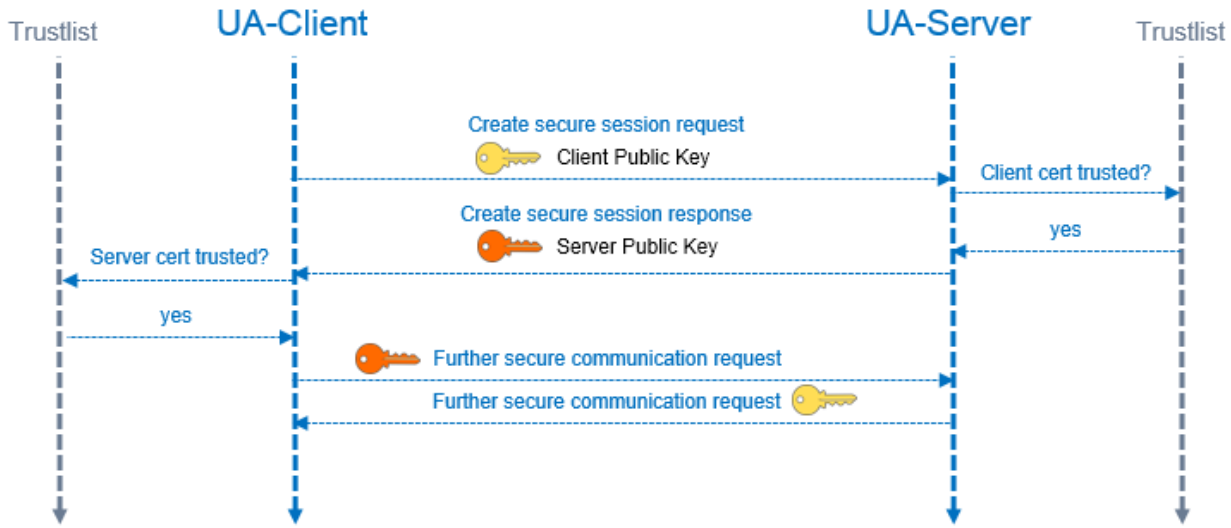
4.2.3.2 Zertifikatsaustausch

Die Kommunikation zwischen einem OPC-UA Client und einem OPC-UA Server kann optional durch Verschlüsselung des Datenverkehrs gesichert werden. Standardmäßig generieren sowohl der TwinCAT OPC-UA Server als auch der TwinCAT OPC-UA Client beim ersten Mal ein maschinenspezifisches, selbstsigniertes Zertifikat zur Authentifizierung.

Wenn Sie in Ihrer Umgebung eine verschlüsselte Verbindung verwenden möchten, müssen Sie ein Vertrauensverhältnis zwischen OPC-UA Server und OPC-UA Client aufbauen.

Dieser Artikel besteht aus den folgenden Kapiteln, die den allgemeinen Austausch von Client- und Server-Zertifikaten beschreiben:

- Anmeldung des OPC-UA Clients beim OPC-UA Server
- Anmeldung des OPC-UA Servers beim OPC-UA Client



Bekanntmachen des OPC-UA Clients beim OPC-UA Server

Möchten Sie einen oder mehrere OPC-UA Clients über Zertifikate am OPC-UA Server authentifizieren, so muss der OPC-UA Server den öffentlichen Schlüsseln der Clients vertrauen. Hierzu muss der öffentliche Schlüssel des jeweiligen Clients in den folgenden Unterordner des OPC-UA Servers kopiert werden:

- TwinCAT 2: **%InstallDir%\UA\PKI\CA\certs**
- TwinCAT 3: **%InstallDir%\Server\PKI\CA\certs**

Dieser Ordner beinhaltet alle Client-Zertifikate, welchen der OPC-UA Server vertraut und mit denen eine authentifizierte Verbindung aufbaut werden kann. Die Client-Zertifikate müssen hierbei im Format **"<beliebigerName>.der"** vorliegen, damit der OPC-UA Server diese verarbeiten kann. Die folgende Anleitung zeigt Ihnen einen Weg, wie man an das Client-Zertifikat gelangt, was für viele Anwendungsfälle wahrscheinlich die einfachste Möglichkeit ist an ein Clientzertifikat zu gelangen. Hierbei bauen Sie, ohne vorher das Clientzertifikat auf den UA-Server kopiert zu haben, eine Verbindung zum UA-Server unter Verwendung eines Security-Endpunkts (z.B. Basic128Rsa15/Sign&Encrypt) auf. Diese Verbindung wird vom UA-Server natürlich abgewiesen, da dieser dem UA-Client zum derzeitigen Zeitpunkt noch nicht vertraut. Der TwinCAT OPC-UA Client würde in diesem Fall z.B. den Fehler 0xE4DD0102 zurückliefern. Nach Abweisen der Verbindungsaufforderung speichert der UA-Server jedoch eine Kopie des Clientzertifikats im folgenden Verzeichnis ab:

- TwinCAT 2: **%InstallDir%\UA\PKI\CA\crl**
- TwinCAT 3: **%InstallDir%\Server\PKI\CA\crl**

Der Name des Zertifikats entspricht hierbei dem "Thumbprint" des Zertifikats und lässt sich somit eindeutig dem UA-Clientzertifikat zuordnen. Diese Datei können Sie nun direkt in den "certs"-Ordner kopieren, damit der UA-Server bei weiteren Verbindungsanforderungen dem UA-Client vertraut und die Verbindung akzeptiert.

- TwinCAT 2: **%InstallDir%\UA\PKI\CA\certs**
- TwinCAT 3: **%InstallDir%\Server\PKI\CA\certs**

Bekanntmachen des OPC-UA Servers beim OPC-UA Client

Abhängig vom verwendeten OPC-UA Client müssen eventuell unterschiedliche Schritte unternommen werden, damit der OPC-UA Client dem OPC-UA Server vertraut. Die folgende Anleitung ist daher nur für den TwinCAT OPC-UA Client gültig.

Der öffentliche Schlüssel des OPC-UA Servers befindet sich als DER-Datei in dem folgenden Verzeichnis:

- TwinCAT 2: **%InstallDir%\UA\PKI\CA\certs**
- TwinCAT 3: **%InstallDir%\Server\PKI\CA\certs**

Der Name der Datei lautet **"Beckhoff_OpcUaServer.der"**. Diese Datei muss beim OPC-UA Client in den folgenden Ordner kopiert werden:

- TwinCAT 2: **%InstallDir%\UA Client\PKI\CA\certs**
- TwinCAT 3: **%InstallDir%\Client\PKI\CA\certs**

Bei einem Drittanbieter OPC-UA Client muss diese Datei in ähnlicher Form bekannt gemacht werden.

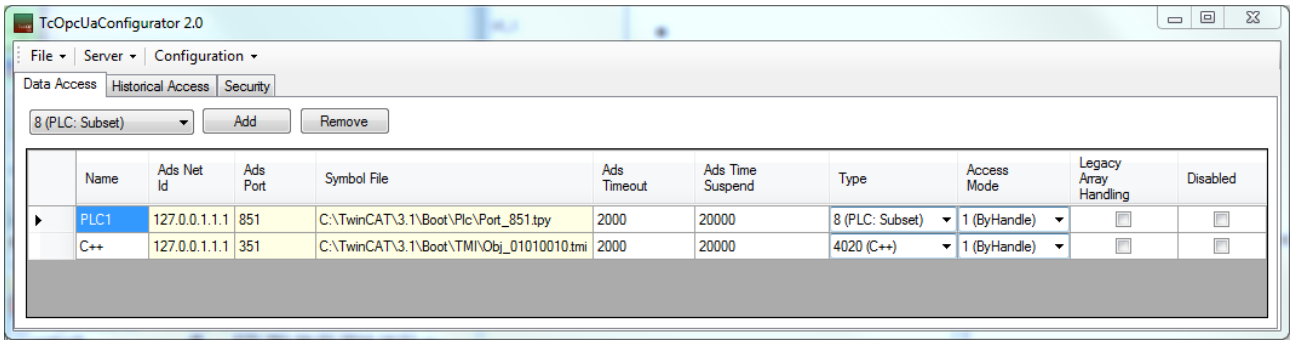
Nun vertraut der entsprechende TwinCAT OPC-UA Client dem UA Server und erlaubt es, eine Verbindung zu diesem herzustellen.

4.3 Konfigurator

4.3.1 Übersicht

Diese Dokumentation soll einen Einblick in den Umgang mit dem TwinCAT OPC-UA Konfigurator geben. Der OPC-UA Konfigurator parametrisiert den TwinCAT OPC-UA Server bzw. ist eine grafische Benutzeroberfläche zur Konfiguration der ServerConfig.xml und bietet hierbei die folgenden Konfigurationsmöglichkeiten:

- Hinzufügen/Entfernen von Geräten zum Data Access (SPS-Laufzeiten, TwinCAT 3 C++-Instanzen, I/O-Tasks)
- Aktivieren/Deaktivieren von Historical Access
- Aktivieren/Deaktivieren von anonymen Zugriffen bzw. Benutzername/Password Authentifizierung
- Neustart oder Shutdown des OPC-UA Servers

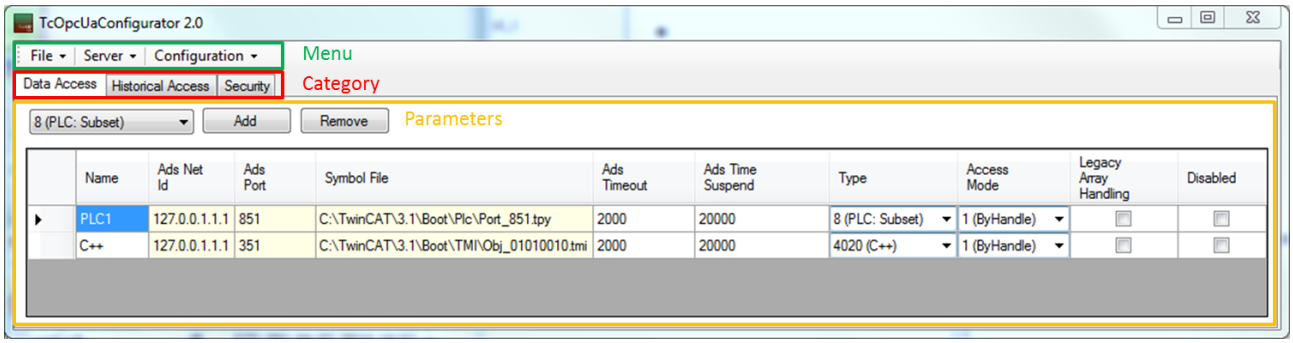


Beim Starten des TwinCAT OPC-UA Konfigurators wird automatisch die derzeit aktive Konfiguration des TwinCAT OPC-UA Servers geladen, welche sich standardmäßig als Datei ServerConfig.xml im Verzeichnis \ %InstallDir%\Server\ befindet. Sie können über das Menü aber auch andere ServerConfig.xml Dateien laden, abspeichern und aktivieren.

Die Benutzeroberfläche des OPC-UA Konfigurators besteht aus den folgenden Bereichen:

- Menü-Bereich: Speichern und Aktivieren von Konfigurationen
- Kategorien-Bereich: Gruppiert die Einzelparameter in verschiedene Kategorien, z.B. Data Access

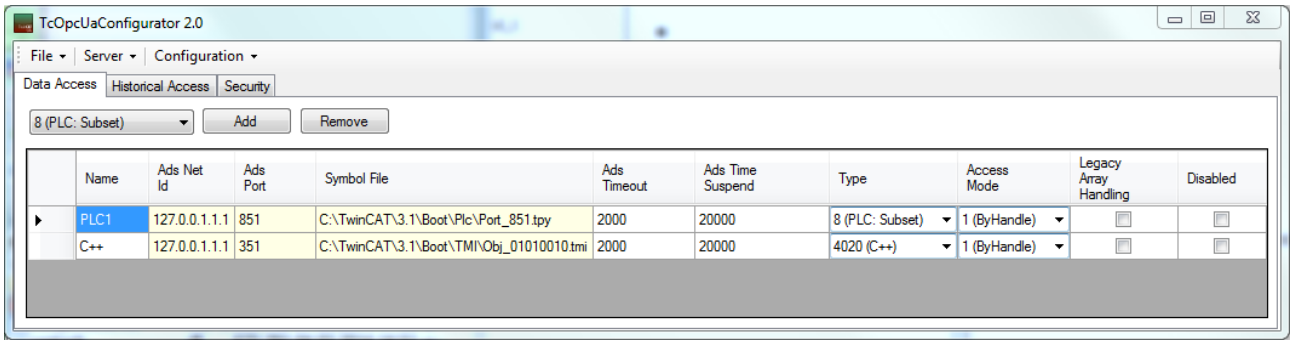
- Parameter-Bereich: Einzelparameter zum Aktivieren/Deaktivieren bestimmter Features



- Hinweis: Bitte konsultieren Sie die Dokumentation der jeweiligen Kategorien (Data Access, Historical Access, …), um die Bedeutung der einzelnen Konfigurationsparameter zu erfahren.

4.3.2 Data Access

Die Registerkarte Data Access stellt die Konfigurationseinstellung für TwinCAT Runtime-Geräte bereit. Hier muss ein Gerät konfiguriert werden, das über OPC-UA verfügbar sein soll.



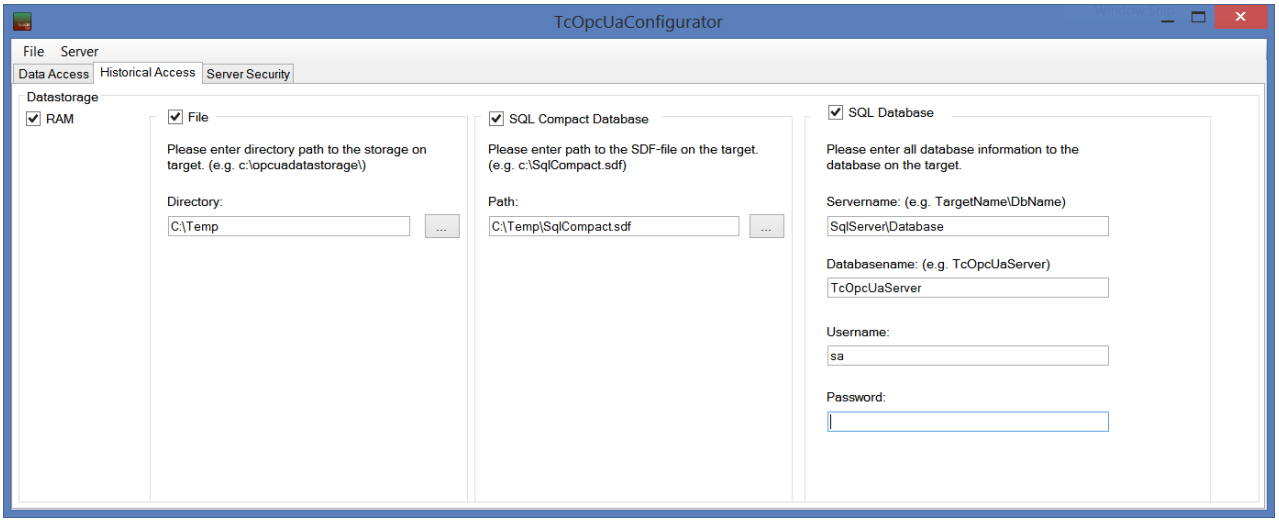
Jedes TwinCAT Runtime-Gerät erfordert die folgenden Einstellungen:

Einstellung	Beschreibung
Name	Anzeige Name der TwinCAT Runtime im OPC-UA Namensraum
AdsNetId	AmsNetId der TwinCAT Runtime. Verwenden Sie 127.0.0.1.1.1 für eine lokale Laufzeit oder die entsprechende AmsNetId eines Remote-Geräts. Beachten Sie bitte, dass die ADS-Routen konfiguriert werden müssen, wenn mit einem Remote-Gerät kommuniziert wird.
AdsPort	AdsPort der TwinCAT Runtime, z. B. 851 für die erste TwinCAT 3 SPS-Laufzeit.
SymbolFile	Symboldatei, die Informationen über alle verfügbaren Variablen und Objektinstanzen der Laufzeit enthält.
AdsTimeout	Timeout-Einstellung, wenn mit der Ziel-Runtime kommuniziert wird. Wir empfehlen, diese Einstellung auf dem Standardwert zu belassen.
AdsTimeSuspend	TimeSuspend-Einstellung, wenn mit der Ziel-Runtime kommuniziert wird. Wir empfehlen, diese Einstellung auf dem Standardwert zu belassen. Der TimeSuspend Timeout wird ausgelöst, wenn eine Ziel-Runtime vorübergehend nicht verfügbar ist. Der OPC-UA Server wird die Kommunikation nach dem TimeSuspend erneut versuchen.
Typ	Typ der Ziel-Runtime, mit der der OPC-UA Server kommunizieren soll, z. B. TwinCAT 3 SPS, TwinCAT 2 SPS, ...
AccessMode	Der Zugriffsmodus ist normalerweise erforderlich, wenn der OPC-UA Server mit einer Laufzeit kommunizieren soll, die auf einem Beckhoff BC/BX Gerät gehostet ist. In diesem Fall muss die entsprechende Einstellung aktiviert werden. In allen anderen Fällen, ist die Standardeinstellung in Ordnung.
LegacyArrayHandling	Wenn veraltete OPC-UA Clients fordern, dass Array-Elemente als separate UA-Knoten verfügbar sind, aktivieren Sie diese Einstellung.
Disabled	Stellen Sie dieses Kontrollkästchen auf Deaktivierung der Laufzeit, so dass sie nicht in der OPC-UA Namensraum veröffentlicht wird.

Weitere Informationen über laufzeitspezifische Parameter erhalten Sie in der OPC-UA Server Dokumentation [▶ 36].

4.3.3 Historical Access

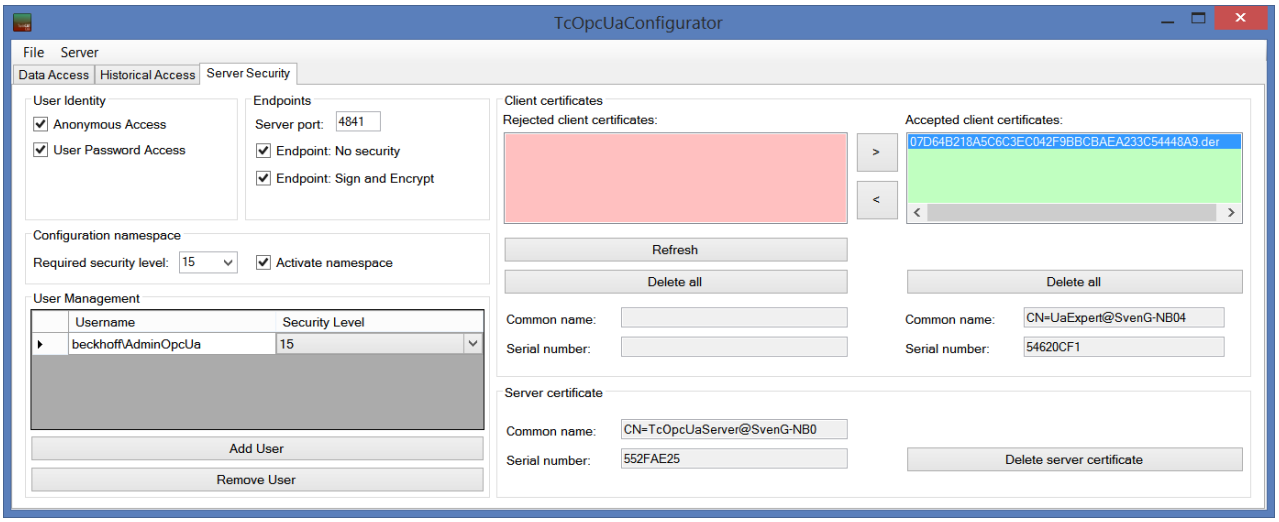
Die Registerkarte Historical Access wird verwendet, um die allgemeinen Einstellungen für Historical Access zu konfigurieren. Hier werden Speichermedien, die für Historical Access verwendet werden sollen, aktiviert und konfiguriert.



Weitere Informationen zu Historical Access erhalten Sie in der OPC-UA Server Dokumentation [▶ 36].

4.3.4 Sicherheit

Die Registerkarte Sicherheit ermöglicht die allgemeinen Einstellungen zu Sicherheit und Endpunktmechanismen des OPC-UA Servers.

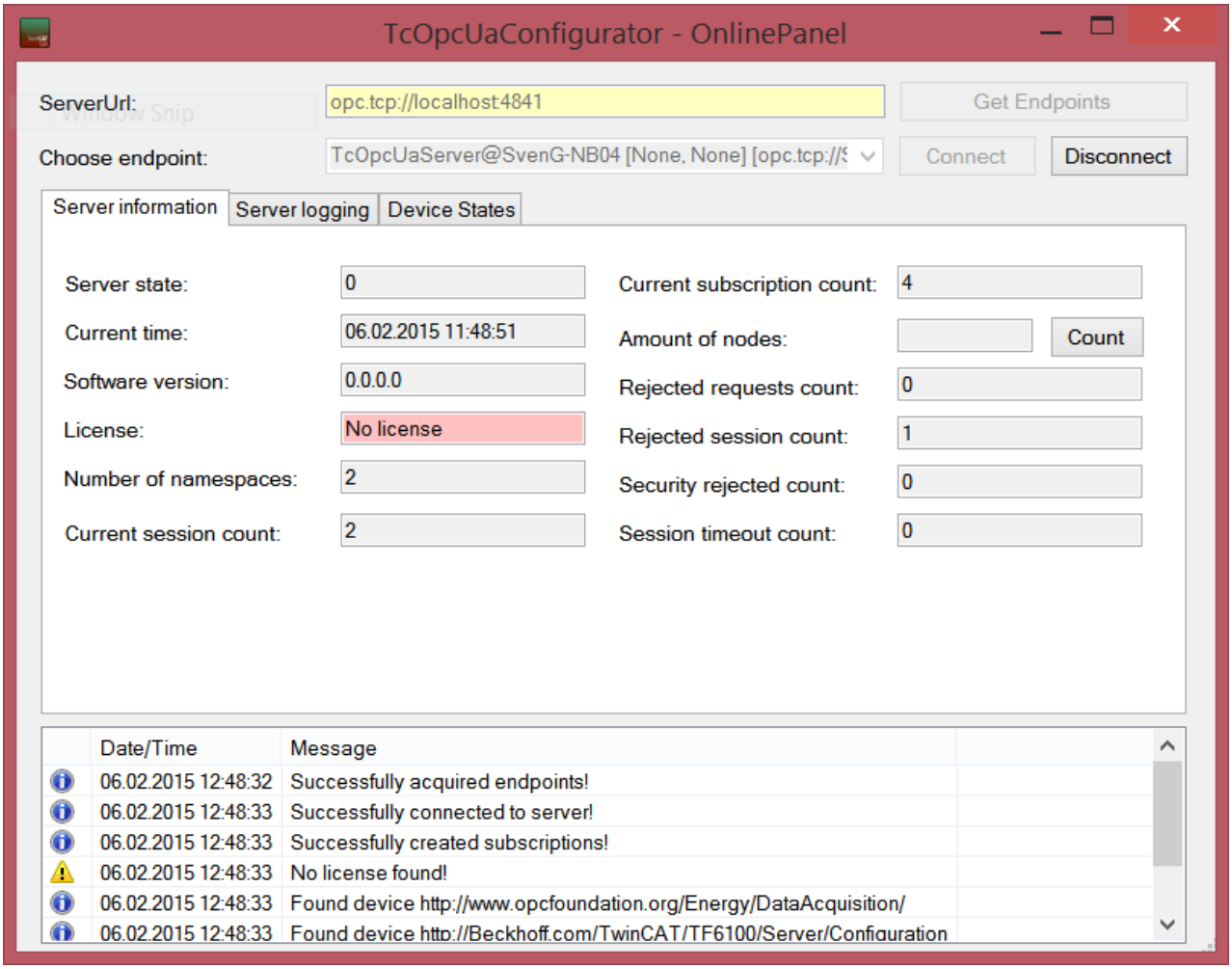


Die folgende Tabelle bietet einen Überblick über alle möglichen Einstellungen:

Einstellung	Beschreibung
Benutzeridentität	Aktivierung oder Deaktivierung einer bestimmten Benutzeridentität, z. B. Deaktivierung des Kontrollkästchens „Anonymous Access“ (Anonymer Zugriff), um eine Benutzerauthentifizierung zu erzwingen.
Endpunkte	Aktivierung oder Deaktivierung bestimmter Endpunkte, z. B. zur Deaktivierung des Endpunkts „No security“ (Keine Sicherheit), um UA-Clients zu zwingen, sich mit dem Endpunkt „Sign and Encrypt“ (Signieren und Verschlüsseln) zu verbinden.
Konfiguration des Namensraums	Ermöglicht die Aktivierung oder Deaktivierung der Konfiguration des Namensraums [▶ 84] und die Einstellung einer spezifischen Sicherheitsstufe, die für den Zugriff dieses Namensraums erforderlich ist (Benutzerauthentifizierung erforderlich).
Benutzerverwaltung	Ermöglich die Konfiguration bestehender Benutzerkonten und stellt eine Zugriffsebene für jedes Konto ein. Zugriffsebenen werden derzeit lediglich zur Konfiguration des Zugriffs auf die Konfiguration des Namensraums [▶ 84] verwendet. Beachten Sie bitte, dass das Benutzerkonto - wenn der Computer, auf dem der OPC-UA Server läuft, ein Mitglied einer Windows-Domäne ist - ebenfalls ein Konto in dieser Domäne sein kann.
Client-Zertifikate	Ermöglicht das Verschieben von OPC-UA Client Zertifikaten zwischen den Vertrauenslisten „Abgelehnt“ und „Akzeptiert“. Hier können Client-Zertifikate einfach identifiziert, ihnen vertraut oder diese abgelehnt werden, wenn die Clients mit einem der sicheren Endpunkte verbunden werden.

4.3.5 Online Panel

Das Online Panel erweitert den TwinCAT OPC-UA Konfigurator um OPC-UA Client Funktionalitäten. Es ermöglicht den Benutzern die lokale oder Fernverbindung mit einem TwinCAT OPC UA Server zwecks Erhalt von Diagnose- oder Verwaltungsfunktionalitäten.



Die folgende Tabelle beschreibt jeden Bereich im Online Panel:

Bereich	Beschreibung
Serverinformationen	Liefert allgemeine Diagnoseinformationen des derzeit mit dem TwinCAT OPC UA Server verbundenen Teilnehmers. Mögliche nützliche Informationen: Lizenzstatus Anzahl der derzeit hergestellten OPC-UA Client Sitzungen Anzahl verfügbarer Knoten (Berechnung kann einige Zeit in Anspruch nehmen) Server-Version
Server-Protokollierung	Stellt Funktionalitäten zur Verfügung, um eine Logdatei auf dem TwinCAT OPC UA Server zu aktivieren. Diese sollte aber nur dann verwendet werden, wenn sie vom Beckhoff-Support empfohlen wurde, da die Erstellung der Logdatei die allgemeine Systemleistung beeinträchtigen kann.
Gerätezustände	Stellt Diagnoseinformationen über jedes auf dem TwinCAT OPC UA Server konfigurierte ADS-Gerät zur Verfügung. Diese Information kann sehr hilfreich sein, wenn die Verbindung mit einem ADS-Gerät überprüft werden muss. Häufig auftretendes Konfigurationsproblem kann z.B. darin bestehen, dass die entsprechende Symboldatei (TPY, TMC) nicht vom TwinCAT OPC UA Server geladen werden kann. Das Device States Panel wird in diesem Falle eine entsprechende Fehlermeldung ausgeben.

4.4 Sample Client

4.4.1 Übersicht

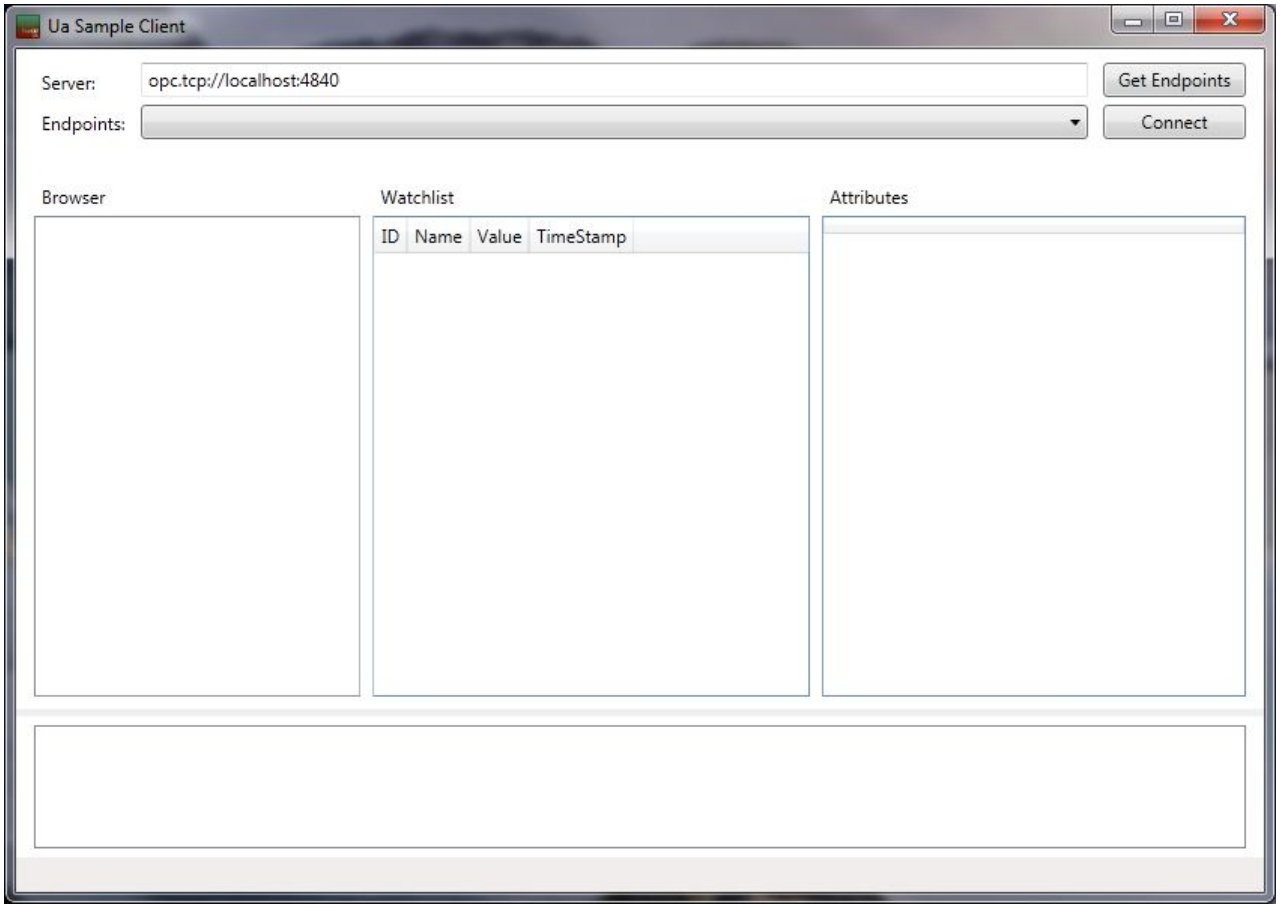
Ab Version 1.6.80 des TwinCAT OPC-UA Servers wird ein kleines "UA Sample Client" genanntes Programm automatisch installiert. Diese Software ermöglicht den Nutzern, den OPC-UA Namensraum zu durchsuchen und die UA-Server Installierung zu testen. Es befindet sich im Windows Startmenü und im Installationsverzeichnis der Supplement/Function. Sie können diesen UA-Client sowohl direkt auf dem UA-Server als auch auf einem Computer in Ihrem Netzwerk ausführen. Derzeit bietet der UA-SampleClient die folgenden Funktionalitäten:

- Verbindung herstellen: Verbindung mit OPC-UA Server herstellen
- Sichere Verbindung herstellen [▶ 110]: Sichere Verbindung mit OPC-UA Server herstellen
- Durchsuchen [▶ 113]: Den UA-Namensraum eines OPC-UA Servers durchsuchen
- Watchliste [▶ 114]: Hinzufügen eines UA Nodes aus dem Namensraum in die Watchliste, die den Wert des Knotens regelmäßig liest.

Bitte beachten:

- Zum Starten der Software einfach die Datei **"UA SampleClient.exe"** ausführen.
- Achten Sie darauf, dass Sie diese Anwendung mit der Option „**Als Administrator ausführen**“ starten.
- Diese Anwendung ist lediglich ein OPC-UA Beispiel-Client. Sie bietet keine anspruchsvollen Funktionalitäten, sondern wurde entwickelt, um den Benutzern eine einfach zu bedienende Schnittstelle zur Verfügung zu stellen, um die ersten Tests beim TwinCAT OPC-UA Server durchzuführen

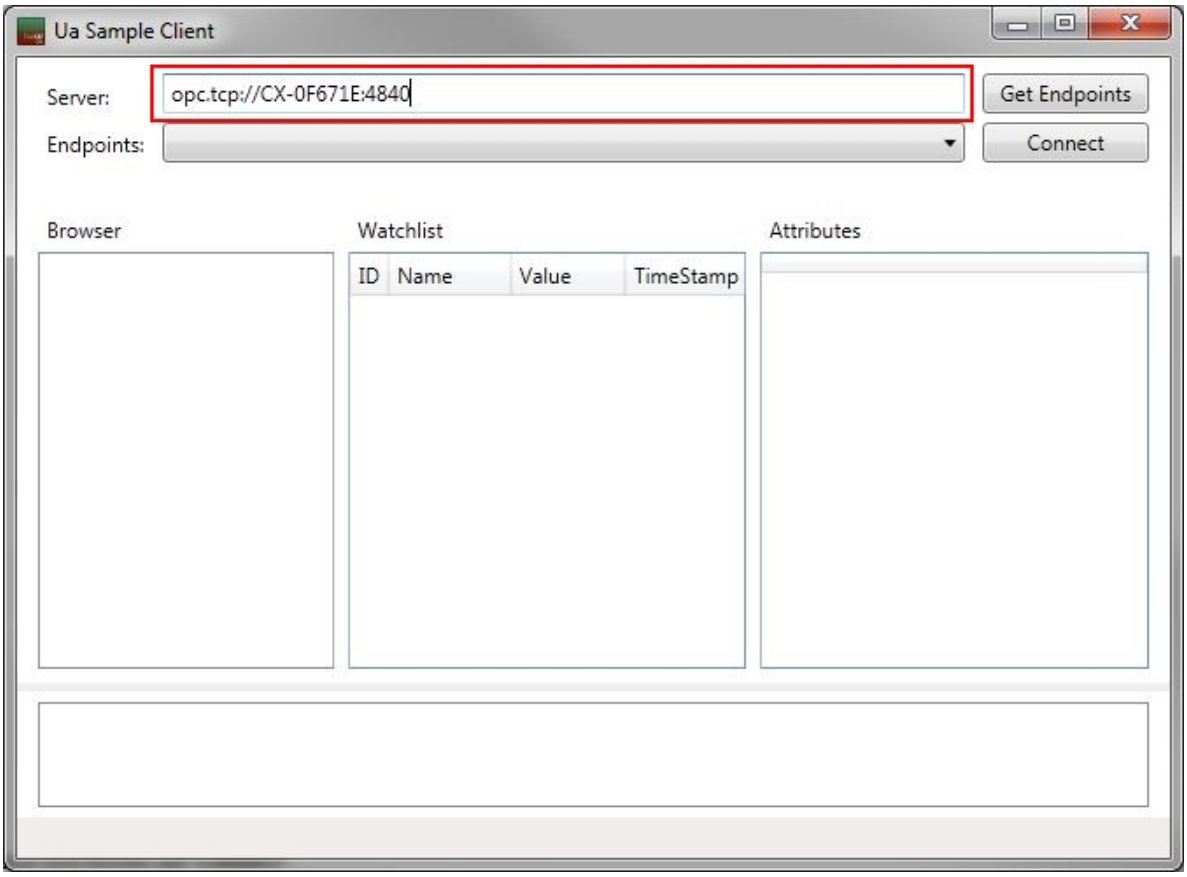
Hinweis	**Wichtiger Hinweis** Der OPC-UA Beispiel-Client verbindet sich zuerst mit einer spezifizierten Server URL. Der Client akquiriert alle Endpunkte des OPC-UA Servers (siehe DropDownBox „Endpunkte”). Die an den Server zurückgesandte Liste enthält dann mehr Informationen über alle verfügbaren Endpunkte, mit denen sich der Client verbinden kann. Jeder Endpunkt kann den Hostnamen des OPC-UA Servers anstatt der IP-Adresse enthalten. Der Client verwendet dann die Informationen aus dem Endpunkt zur Verbindung mit dem Server. Wenn die Namenslösung nicht im Netzwerk des Benutzers funktioniert, kann der Client die Verbindung nicht herstellen. Wenn der Endpunkt, mit dem sich der Client verbinden soll, den Hostnahmen des Servers enthält, stellen Sie sicher, dass die Namenslösung in Ihrem Netzwerk funktioniert und dass der Hostname auf dem Server erreichbar ist!



4.4.2 Sichere Verbindung mit OPC-UA Server herstellen

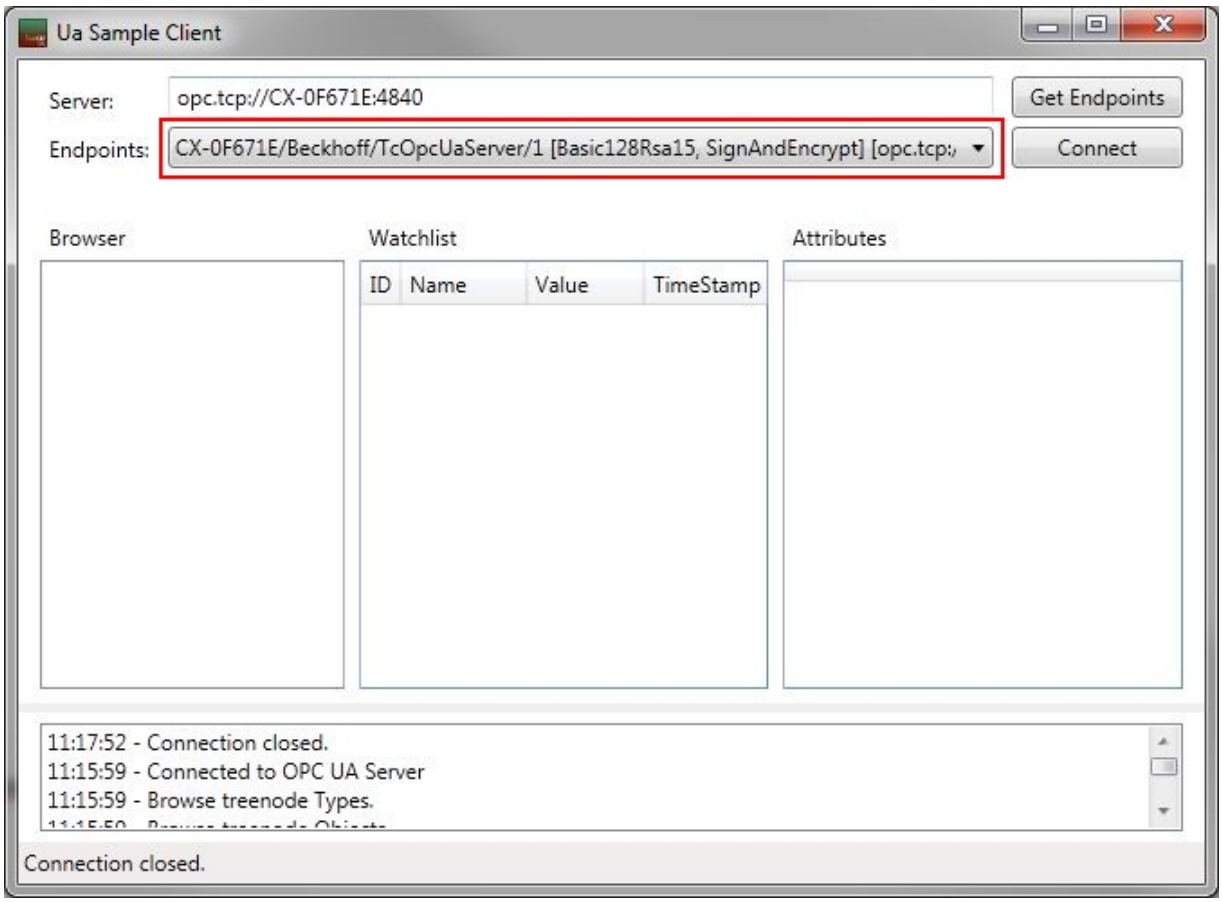
1. Geben Sie in der oberen Textbox des UA-SampleClient die URL eines OPC-UA Servers ein.
2. Klicken Sie danach auf den Button "**Get Endpoints**".

⇨ Die vom UA-Server bereitgestellten Endpunkte werden daraufhin in der DropDown-Box "**Endpoints**" angezeigt.

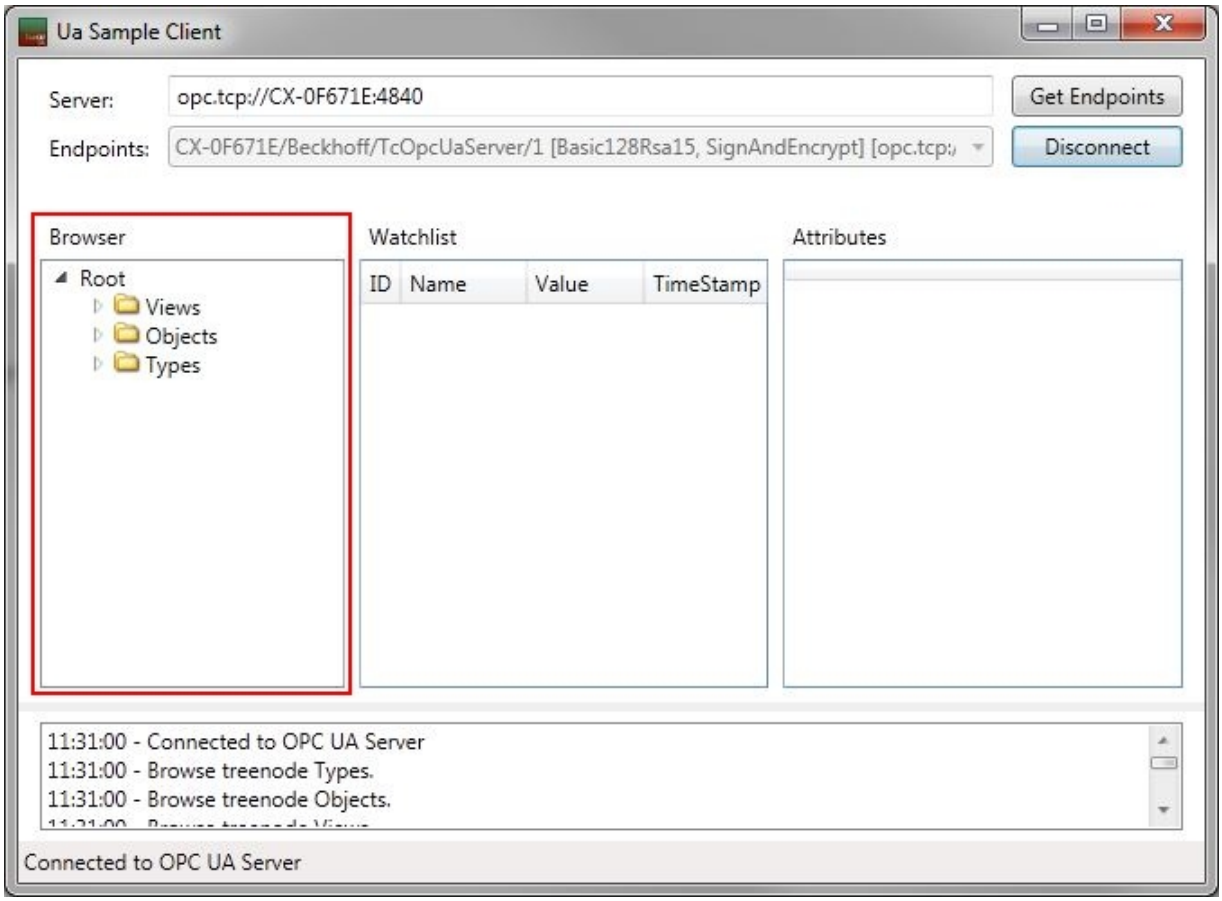


3. In diesem Beispiel wählen Sie bitte den Eintrag "**<SomeName>/Beckhoff/TcOpcUaServer/1 [Basic128Rsa15, SignAndEncrypt] [opc.tcp://<SomeName>:4840]**" aus und klicken auf "**Connect**".
   HINWEIS! Sie müssen den Public Key vom Zertifikat des UA-SampleClients auf den UA-Server kopieren, damit dieser dem SampleClient "vertraut". Andernfalls wird der Verbindungsversuch

über den sicheren Kanal vom UA-Server abgewiesen ("BadSecureChannelClosed"). Weitere Informationen über das Zertifikatsmanagement beim TwinCAT OPC-UA Server finden Sie hier [▶ 102].

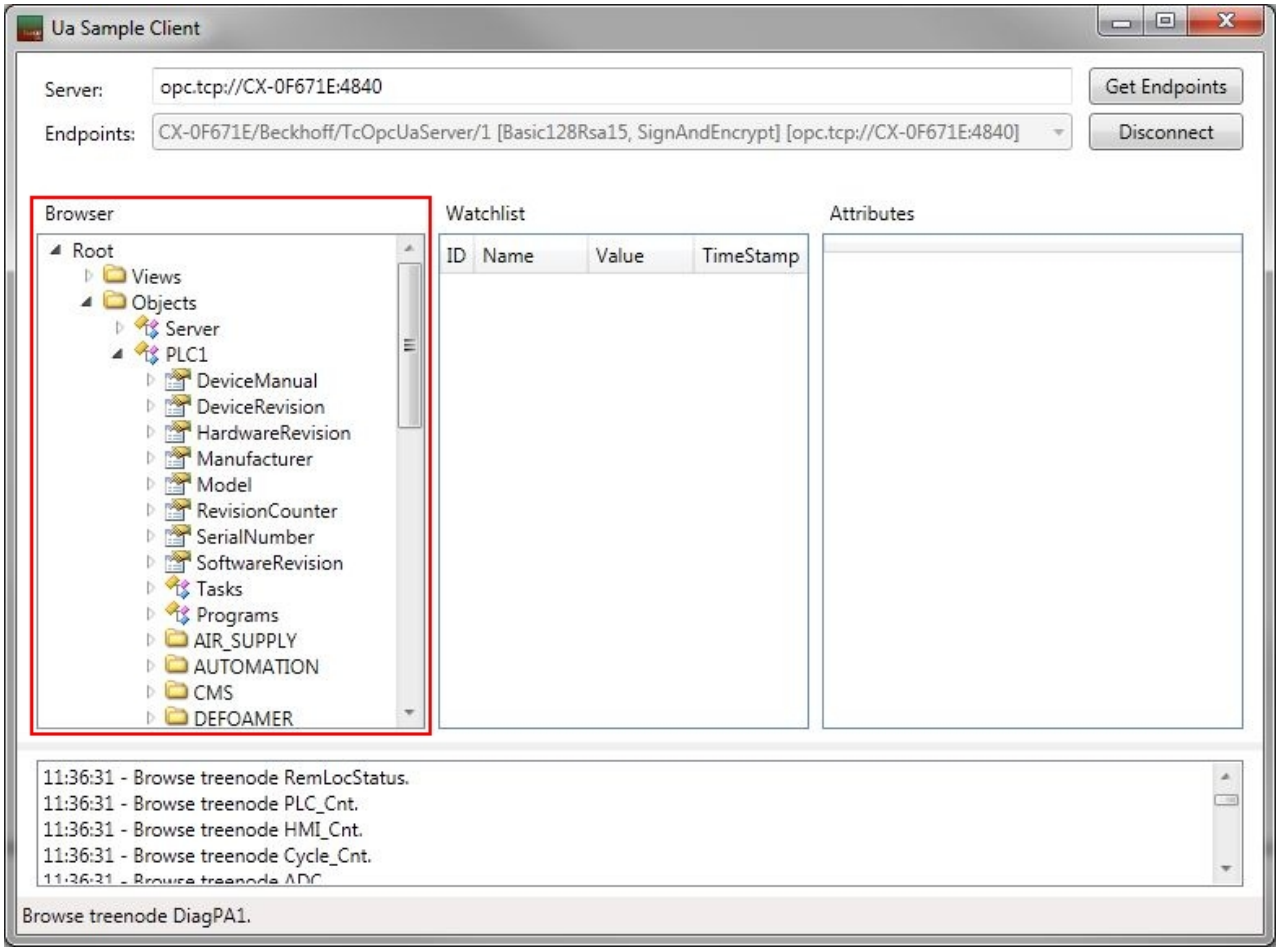


4. Über den "Browser" in der linken Hälfte des Fensters können Sie nun durch den UA-Namensraum navigieren.



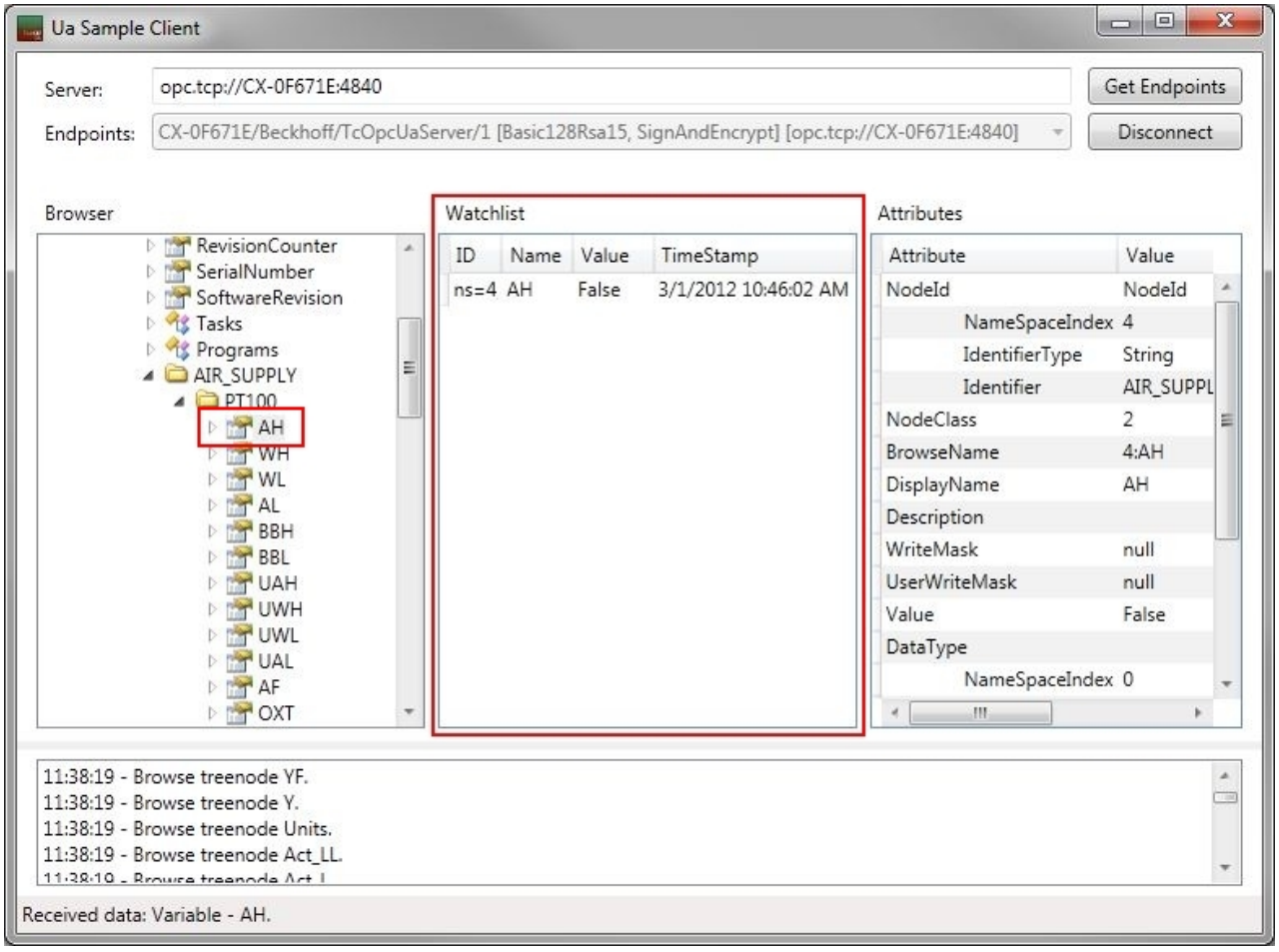
4.4.3 Im UA-Namensraum browsen

Über den "Browser" in der linken Hälfte des UA-SampleClient können Sie nach erfolgreichem Aufbau einer Verbindung durch den UA-Namensraum navigieren. Unterhalb des Knotens "**PLC1**" finden Sie das momentan laufende SPS-Programm und können die dort deklarierten und für UA freigegebenen Variablen anzeigen lassen. Weitere Informationen zur Konfiguration des UA-Servers und das Freigeben von SPS-Variablen für OPC-UA finden Sie hier.



4.4.4 Watchliste verwenden

Sie können SPS-Variablen aus dem UA-Namensraum in eine Watchlist einfügen, zum Beispiel um deren Werte zyklisch von dem UA-SampleClient lesen zu lassen. Hierzu markieren Sie eine Variable, klicken sie mit der rechten Maustaste an und wählen "**Add to Watchlist**". Die Variable wird daraufhin in die Watchlist übernommen und deren Werte werden automatisch zyklisch aus der SPS ausgelesen.

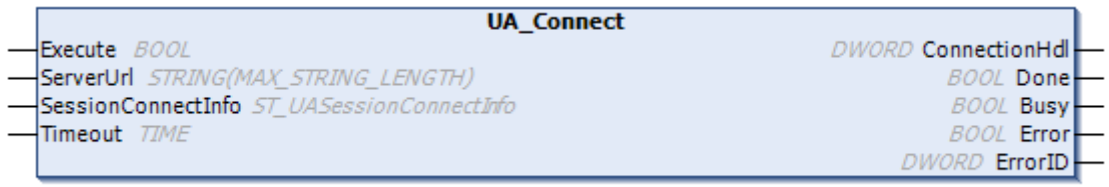

Ua Sample Client
Server:
opc.tcp://CX-0F671E:4840
Get Endpoints
Endpoints:
CX-0F671E/Beckhoff/TcOpcUaServer/1 [Basic128Rsa15, SignAndEncrypt] [opc.tcp://CX-0F671E:4840]
Disconnect
Browser
RevisionCounter
SerialNumber
SoftwareRevision
Tasks
Programs
AIR_SUPPLY
PT100
AH
WH
WL
AL
BBH
BBL
UAH
UWH
UWL
UAL
AF
OXT
Watchlist
ID
Name
Value
TimeStamp
ns=4
AH
False
3/1/2012 10:46:02 AM
Attributes
Attribute
Value
NodeId
NodeId
NameSpaceIndex
4
IdentifierType
String
Identifier
AIR_SUPPL
NodeClass
2
BrowseName
4:AH
DisplayName
AH
Description
WriteMask
null
UserWriteMask
null
Value
False
DataType
NameSpaceIndex
0
11:38:19 - Browse treenode YF.
11:38:19 - Browse treenode Y.
11:38:19 - Browse treenode Units.
11:38:19 - Browse treenode Act_LL.
Received data: Variable - AH.

5 SPS-Bibliotheken

5.1 Tcx_PLCopen_OpcUa

5.1.1 Funktionsbausteine

5.1.1.1 UA_Connect



Dieser Funktionsbaustein stellt eine OPC-UA-Remote-Verbindung zu einem anderen OPC-UA Server her, der via ServerUrl und SessionConnectInfo spezifiziert wird. Der Funktionsbaustein gibt ein Verbindungshandle zurück, der für andere Funktionsbausteine, z.B. UA_Read verwendet werden kann.

VAR_INPUT

```
VAR_INPUT
    Execute            : BOOL;
    ServerUrl          : STRING(MAX_STRING_LENGTH);
    SessionConnectInfo : ST_UASessionConnectInfo;
    Timeout            : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

ST_UASessionConnectInfo [▶ 129]

Execute : Der Befehl wird durch eine steigende Flanke an diesem Eingang ausgelöst.

ServerUrl : OPC-UA Server URL. d.h. 'opc.tcp://172.16.3.207:4840' oder 'opc.tcp://CX_0193BF:4840'.

SessionConnectInfo : Information für Verbindung. Siehe ST_UASessionConnectInfo. [▶ 129]

Timeout : Zeit bis zum Abbruch der Function. DEFAULT_ADS_TIMEOUT ist eine globale Konstante, gesetzt auf 5 Sekunden.
Achtung: ST_UASessionConnectInfo. [▶ 129]tSessionTimeout muss kürzer sein als dieses Timeout.

VAR_OUTPUT

```
VAR_OUTPUT
    ConnectionHdl : DWORD;
    Done          : BOOL;
    Busy          : BOOL;
    Error         : BOOL;
    ErrorID       : DWORD;
END_VAR
```

ConnectionHdl : OPC-UA Verbindungshandle.

Done : Schaltet auf TRUE, wenn der Funktionsbaustein ausgeführt wurde.

Busy: Dieser Ausgang ist TRUE, bis der Baustein einen Befehl ausführt hat, maximal für die Dauer des 'Timeout' am Eingang. Solange Busy = TRUE akzeptieren die Eingänge keinen neuen Befehl. Es wird nicht die Zeit der Verbindung sondern die Empfangszeit überwacht.

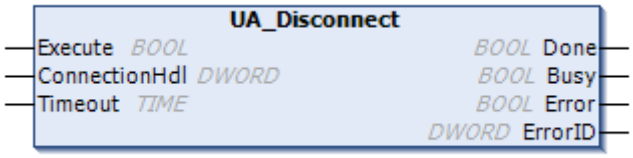
Error : Dieser Ausgang schaltet auf TRUE, wenn bei der Ausführung eines Befehls ein Fehler auftritt. Der befehlsspezifische Fehlercode ist in 'ErrorID' enthalten.

ErrorID : Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT 3.1	Win32, Win64, CE-X86, CE-ARM	Tc3_PLCopen_OpcUa
TwinCAT 2.11 R3 Build 2245	Win32, CE-X86, CE-ARM	Tc2_PLCopen_OpcUa

5.1.1.2 UA_Disconnect



Dieser Funktionsbaustein schließt eine OPC-UA-Remote-Verbindung zu einem anderen OPC-UA Server. Die Verbindung wird über ihr Verbindungshandle spezifiziert.

VAR_INPUT

```
VAR_INPUT
    Execute           : BOOL;
    ConnectionHdl     : DWORD;
    Timeout           : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

Execute : Der Befehl wird durch eine steigende Flanke an diesem Eingang ausgelöst.

ConnectionHdl : Verbindungshandle, das vorher vom Funktionsbaustein UA_Connect [▶ 116] ausgegeben wurde.

Timeout : Zeit bis zum Abbruch der Function. DEFAULT_ADS_TIMEOUT ist eine globale Konstante, gesetzt auf 5 Sekunden.

VAR_OUTPUT

```
VAR_OUTPUT
    Done        : BOOL;
    Busy        : BOOL;
    Error       : BOOL;
    ErrorID     : DWORD;
END_VAR
```

Done : Schaltet auf TRUE, wenn der Funktionsbaustein ausgeführt wurde.

Busy : Dieser Ausgang ist TRUE, bis der Baustein einen Befehl ausführt hat, maximal für die Dauer des 'Timeout' am Eingang. Solange Busy = TRUE akzeptieren die Eingänge keinen neuen Befehl. Es wird nicht die Zeit der Verbindung sondern die Empfangszeit überwacht.

Error : Dieser Ausgang schaltet auf TRUE, wenn bei der Ausführung eines Befehls ein Fehler auftritt. Der befehlsspezifische Fehlercode ist in ‚nErrID' enthalten.

ErrorID : Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT 3.1	Win32, Win64, CE-X86, CE-ARM	Tc3_PLCopen_OpcUa
TwinCAT 2.11 R3 Build 2245	Win32, CE-X86, CE-ARM	Tc2_PLCopen_OpcUa

5.1.1.3 UA_GetNamespaceIndex

Dieser Funktionsbaustein erfasst den Namespace Index für einen Namespace URI. Der Namespace-Index wird für die Identifizierung von Symbolen benötigt, z.B. wenn die Funktionsbausteine UA_Read [▶ 124]oder UA_Write [▶ 125]genutzt werden.

VAR_INPUT

```
VAR_INPUT
    Execute       : BOOL;
    ConnectionHdl : DWORD;
    NamespaceUri  : STRING(MAX_STRING_LENGTH);
    Timeout       : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

Execute : Der Befehl wird durch eine steigende Flanke an diesem Eingang ausgelöst.

ConnectionHdl : Verbindungshandle, das vorher vom Funktionsbaustein UA_Connect [▶ 116] ausgegeben wurde.

NamespaceUri : Namensraum URI, der aufgelöst werden soll. Beim TwinCAT OPC-UA Server ist das für die erste SPS-Laufzeit "PLC1".

Timeout : Zeit bis zum Abbruch der Function. DEFAULT_ADS_TIMEOUT ist eine globale Konstante, gesetzt auf 5 Sekunden.

VAR_OUTPUT

```
VAR_OUTPUT
    NamespaceIndex : UINT;
    Done           : BOOL;
    Busy           : BOOL;
    Error          : BOOL;
    ErrorID        : DWORD;
END_VAR
```

NamespaceIndex : Namespace-Index des gegebenen Namensraum URI. Dieser kann in anderen Funktionsbausteinen verwendet werden, z.B. UA_Read oder UA_Write.

Done : Schaltet auf TRUE, wenn der Funktionsbaustein ausgeführt wurde.

Busy: Dieser Ausgang ist TRUE, bis der Baustein einen Befehl ausführt hat, maximal für die Dauer des 'Timeout' am Eingang. Solange Busy = TRUE akzeptieren die Eingänge keinen neuen Befehl. Es wird nicht die Zeit der Verbindung sondern die Empfangszeit überwacht.

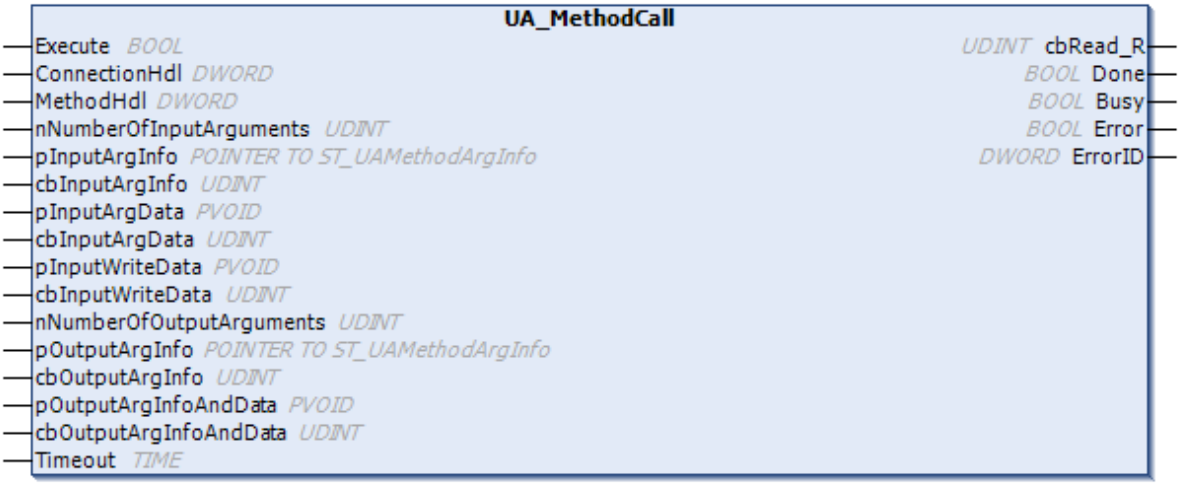
Error : Dieser Ausgang schaltet auf TRUE, wenn bei der Ausführung eines Befehls ein Fehler auftritt. Der befehlsspezifische Fehlercode ist in 'ErrorID' enthalten.

ErrorID : Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT 3.1	Win32, Win64, CE-X86, CE-ARM	Tc3_PLCopen_OpcUa
TwinCAT 2.11 R3 Build 2245	Win32, CE-X86, CE-ARM	Tc2_PLCopen_OpcUa

5.1.1.4 UA_MethodCall



Dieser Funktionsbaustein ruft eine Methode auf einem Remote-UA-Server auf. Die Methode wird durch eine Verbindung und ein Methodenhandle bestimmt. Erstere kann durch UA_Connect [▶ 116] und Letzterer durch UA_MethodGetHandle [▶ 120] abgefragt werden.

VAR_INPUT

```
VAR_INPUT
    Execute                 : BOOL;
    ConnectionHdl           : DWORD;
    MethodHdl               : DWORD;
    nNumberOfInputArguments : UDINT;
    pInputArgInfo           : POINTER TO ST_UAMethodArgInfo;
    cbInputArgInfo          : UDINT;
    pInputArgData           : PVOID;
    cbInputArgData          : UDINT;
    pInputWriteData         : PVOID;
    cbInputWriteData        : UDINT;
    nNumberOfOutputArguments : UDINT;
    pOutputArgInfo          : POINTER TO ST_UAMethodArgInfo;
    cbOutputArgInfo         : UDINT;
    pOutputArgInfoAndData   : PVOID;
    cbOutputArgInfoAndData  : UDINT;
    Timeout                 : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

ST_UAMethodArgInfo [▶ 130] ST_UAMethodArgInfo [▶ 130]

Execute: Der Befehl wird durch eine steigende Flanke an diesem Eingang ausgelöst.

ConnectionHdl: Verbindungshandle, das vorher vom Funktionsbaustein UA_Connect [▶ 116] ausgegeben wurde.

MethodHdl: Methodenhandle, zuvor vom Funktionsbaustein UA_MethodGetHandle [▶ 120] ausgegeben.

nNumberOfInputArguments: Anzahl Eingabeparameter.

pInputArgInfo: Zeigt auf die Pufferadresse, wo Eingabeparameterinformationen in Form eines Arrays ST_UAMethodArgInfo [▶ 130] hinterlegt sind.

cbInputArgInfo: Größe des Puffers, wo die Eingabeparameterinformation hinterlegt ist.

pInputArgData: Zeigt auf die Pufferadresse, wo Eingabeparameter (konstanter Länge) hinterlegt sind.

cbInputArgData: Größe des Eingabepuffers, wo Eingabeparameter (mit konstanter Länge) hinterlegt sind.

pInputWriteData: Zeiger auf Pufferadresse, wo Eingabeparameter (dynamischer Länge) hinterlegt sind.

cbInputWriteData: Größe des Eingabepuffers, wo Eingabeparameter (mit dynamischer Länge) hinterlegt sind.

nNumberOfOutputArguments: Anzahl Ausgabeparameter.

pOutputArgInfo: Zeigt auf die Pufferadresse, wo Ausgabeparameterinformationen als Array ST_UAMethodArgInfo [▶ 130] hinterlegt sind;
Für die Bestimmung des Zielspeichers der einzelnen Ausgabeparameter ist nLenData erforderlich.
Die anderen Elemente können so gesetzt werden, dass eine Typprüfung der zurückgegebenen Parameter erfolgt oder undefiniert bleiben.

cbOutputArgInfo : Größe des Puffers, wo die Ausgabeparameterinformation hinterlegt ist.

pOutputArgInfoAndData: Zeigt auf die Adresse des Puffers, wo die Output-Parameter als BYTE-Array gespeichert werden sollen. Das BYTE Array enthält die Anzahl der Output-Parameter als DINT, 4 reservierte Bytes und Parameterinformationen als ARRAY OF ST_UAMethodArgInfo (mit der Länge der Output-Parameter) gefolgt von reinen Daten. Bitte beachten Sie, dass die Daten als 1byte Alignment gepackt sind.

cbOutputArgInfoAndData: Größe des Puffers, in dem die Output-Parameter als BYTE-Array gespeichert werden sollen.

Timeout: Zeit bis zum Abbruch der Function. DEFAULT_ADS_TIMEOUT ist eine globale Konstante, gesetzt auf 5 Sekunden.

VAR_OUTPUT

```
VAR_OUTPUT
    cbRead_R      : UDINT;
    Done          : BOOL;
    Busy          : BOOL;
    Error         : BOOL;
    ErrorID       : UDINT;
END_VAR
```

cbRead_R : Zählung aller empfangener Bytes.

Done : Schaltet auf TRUE, wenn der Funktionsbaustein ausgeführt wurde.

Busy : Dieser Ausgang ist TRUE, bis der Baustein einen Befehl ausführt hat, maximal für die Dauer des 'Timeout' am Eingang. Solange Busy = TRUE akzeptieren die Eingänge keinen neuen Befehl. Es wird nicht die Zeit der Verbindung sondern die Empfangszeit überwacht.

Error : Dieser Ausgang schaltet auf TRUE, wenn bei der Ausführung eines Befehls ein Fehler auftritt. Der befehlsspezifische Fehlercode ist in ‚nErrID' enthalten.

ErrorID : Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT 3.1	Win32, Win64, CE-X86, CE-ARM	Tc3_PLCopen_OpcUa

5.1.1.5 UA_MethodGetHandle

Dieser Funktionsbaustein erfasst ein Handle für eine UA-Methode, das dann für den Aufruf einer Methode über UA_MethodCall [▶ 119] verwendet werden kann.

VAR_INPUT

```
VAR_INPUT
    Execute          : BOOL;
    ConnectionHdl    : DWORD;
    ObjectNodeID     : ST_UANodeID;
    MethodNodeID     : ST_UANodeID;
    Timeout          : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

ST_UANodeID [▶ 131] ST_UANodeID [▶ 131]

Execute : Der Befehl wird durch eine steigende Flanke an diesem Eingang ausgelöst.

ConnectionHdl : Verbindungshandle, das vorher vom Funktionsbaustein UA_Connect [▶ 116] ausgegeben wurde.

ObjectNodeID : Objektknoten-ID der aufzurufenden Methode.

MethodNodeID : Methoden-Knoten-ID der aufzurufenden Methode. Entspricht dem ID-Attritbut im UA Namensraum.

Timeout : Zeit bis zum Abbruch der Function. DEFAULT_ADS_TIMEOUT ist eine globale Konstante, gesetzt auf 5 Sekunden.

VAR_OUTPUT

```
VAR_OUTPUT
    MethodHdl    : DWORD;
    Done         : BOOL;
    Busy         : BOOL;
    Error        : BOOL;
    ErrorID      : UDINT;
END_VAR
```

MethodHdl : Gibt Methodenhandle zurück, das für den Aufruf einer Methode über UA_MethodCall [▶ 119] verwendet werden kann.

Done : Schaltet auf TRUE, wenn der Funktionsbaustein ausgeführt wurde.

Busy : Dieser Ausgang ist TRUE, bis der Baustein einen Befehl ausführt hat, maximal für die Dauer des 'Timeout' am Eingang. Solange Busy = TRUE akzeptieren die Eingänge keinen neuen Befehl. Es wird nicht die Zeit der Verbindung sondern die Empfangszeit überwacht.

Error : Dieser Ausgang schaltet auf TRUE, wenn bei der Ausführung eines Befehls ein Fehler auftritt. Der befehlsspezifische Fehlercode ist in ‚nErrID' enthalten.

ErrorID : Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT 3.1	Win32, Win64, CE-X86, CE-ARM	Tc3_PLCopen_OpcUa

5.1.1.6 UA_MethodReleaseHandle

Dieser Funktionsbaustein gibt das spezifizierte Methodenhandle frei.

VAR_INPUT

```
VAR_INPUT
    Execute           : BOOL;
    ConnectionHdl     : DWORD;
    MethodHdl         : DWORD;
    Timeout           : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

Execute : Der Befehl wird durch eine steigende Flanke an diesem Eingang ausgelöst.

ConnectionHdl : Verbindungshandle, das vorher vom Funktionsbaustein UA_Connect [▶ 116] ausgegeben wurde.

MethodHdl : Methodenhandle, zuvor vom Funktionsbaustein UA_MethodGetHandle [▶ 120] ausgegeben.

Timeout : Zeit bis zum Abbruch der Function. DEFAULT_ADS_TIMEOUT ist eine globale Konstante, gesetzt auf 5 Sekunden.

VAR_OUTPUT

```
VAR_OUTPUT
    Done        : BOOL;
    Busy        : BOOL;
    Error       : BOOL;
    ErrorID     : UDINT;
END_VAR
```

Done : Schaltet auf TRUE, wenn der Funktionsbaustein ausgeführt wurde.

Busy : Dieser Ausgang ist TRUE, bis der Baustein einen Befehl ausführt hat, maximal für die Dauer des 'Timeout' am Eingang. Solange Busy = TRUE akzeptieren die Eingänge keinen neuen Befehl. Es wird nicht die Zeit der Verbindung sondern die Empfangszeit überwacht.

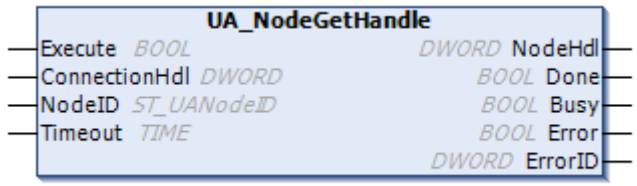
Error : Dieser Ausgang schaltet auf TRUE, wenn bei der Ausführung eines Befehls ein Fehler auftritt. Der befehlsspezifische Fehlercode ist in ‚nErrID' enthalten.

ErrorID : Enthält den befehlsspezifischen ADS Fehlercode des zuletzt ausgeführten Befehls.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT 3.1	Win32, Win64, CE-X86, CE-ARM	Tc3_PLCopen_OpcUa

5.1.1.7 UA_NodeGetHandle



Dieser Funktionsbaustein fragt ein Knotenhandle für ein gegebenes Symbol im UA-Namensraum ab. Das Symbol wird durch ein Verbindungshandle und seine Knoten-ID spezifiziert.

VAR_INPUT

```
VAR_INPUT
    Execute       : BOOL;
    ConnectionHdl : DWORD;
```

```
    NodeID        : ST_UANodeID;
    Timeout       : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

Execute : Der Befehl wird durch eine steigende Flanke an diesem Eingang ausgelöst.

ConnectionHdl : Verbindungshandle, das vorher vom Funktionsbaustein UA_Connect [▶ 116] ausgegeben wurde.

Node ID : Knotenhandle, der zuvor vom Funktionsbaustein UA_NodeGetHandle [▶ 122] ausgegeben wurde.

Timeout : Zeit bis zum Abbruch der Function. DEFAULT_ADS_TIMEOUT ist eine globale Konstante, gesetzt auf 5 Sekunden.

VAR_OUTPUT

```
VAR_OUTPUT
    NodeHdl      : DWORD;
    Done         : BOOL;
    Busy         : BOOL;
    Error        : BOOL;
    ErrorID      : DWORD;
END_VAR
```

NodeHdl : Knotenhandle, das für andere Funktionsbausteine verwendet werden kann, z.B. UA_Read oder UA_Write.

Done : Schaltet auf TRUE, wenn der Funktionsbaustein ausgeführt wurde.

Busy : Dieser Ausgang ist TRUE, bis der Baustein einen Befehl ausführt hat, maximal für die Dauer des 'Timeout' am Eingang. Solange Busy = TRUE akzeptieren die Eingänge keinen neuen Befehl. Es wird nicht die Zeit der Verbindung sondern die Empfangszeit überwacht.

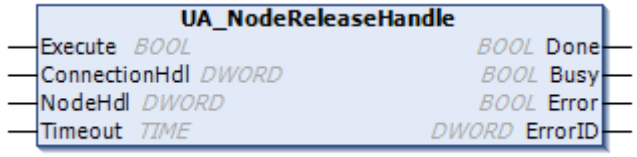
Error : Dieser Ausgang schaltet auf TRUE, wenn bei der Ausführung eines Befehls ein Fehler auftritt. Der befehlsspezifische Fehlercode ist in 'ErrorID' enthalten.

ErrorID : Enthält den befehlsspezifischen ADS Fehlercode des zuletzt ausgeführten Befehls.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT 3.1	Win32, Win64, CE-X86, CE-ARM	Tc3_PLCopen_OpcUa
TwinCAT 2.11 R3 Build 2245	Win32, CE-X86, CE-ARM	Tc2_PLCopen_OpcUa

5.1.1.8 UA_NodeReleaseHandle



Dieser Funktionsbaustein gibt ein Knotenhandle frei.

VAR_INPUT

```
VAR_INPUT
    Execute       : BOOL;
    ConnectionHdl : DWORD;
    NodeHdl       : DWORD;
    Timeout       : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

Execute : Der Befehl wird durch eine steigende Flanke an diesem Eingang ausgelöst.

ConnectionHdl : Verbindungshandle, das vorher vom Funktionsbaustein UA_Connect [▶ 116] ausgegeben wurde.

NodeHdl : Freizugebendes Knotenhandle.

Timeout : Zeit bis zum Abbruch der Function. DEFAULT_ADS_TIMEOUT ist eine globale Konstante, gesetzt auf 5 Sekunden.

VAR_OUTPUT

```
VAR_OUTPUT
    Done        : BOOL;
    Busy        : BOOL;
    Error       : BOOL;
    ErrorID     : DWORD;
END_VAR
```

Done : Schaltet auf TRUE, wenn der Funktionsbaustein ausgeführt wurde.

Busy : Dieser Ausgang ist TRUE, bis der Baustein einen Befehl ausführt hat, maximal für die Dauer des 'Timeout' am Eingang. Solange Busy = TRUE akzeptieren die Eingänge keinen neuen Befehl. Es wird nicht die Zeit der Verbindung sondern die Empfangszeit überwacht.

Error : Dieser Ausgang schaltet auf TRUE, wenn bei der Ausführung eines Befehls ein Fehler auftritt. Der befehlsspezifische Fehlercode ist in 'ErrorID' enthalten.

ErrorID : Enthält den befehlsspezifischen ADS Fehlercode des zuletzt ausgeführten Befehls.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT 3.1	Win32, Win64, CE-X86, CE-ARM	Tc3_PLCopen_OpcUa
TwinCAT 2.11 R3 Build 2245	Win32, CE-X86, CE-ARM	Tc2_PLCopen_OpcUa

5.1.1.9 UA_Read

Dieser Funktionsbaustein liest Werte aus einem gegebenen Knoten- und Verbindungshandle.

VAR_INPUT

```
VAR_INPUT
    Execute           : BOOL;
    ConnectionHdl     : DWORD;
    NodeHdl           : DWORD;
    stNodeAddInfo     : ST_UANodeAdditionalInfo;
    pVariable         : PVOID;
    cbData            : UDINT;
    Timeout           : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

Execute : Der Befehl wird durch eine steigende Flanke an diesem Eingang ausgelöst.

ConnectionHdl : Verbindungshandle, das vorher vom Funktionsbaustein UA_Connect [▶ 116] ausgegeben wurde.

NodeHdl : Knotenhandle, der zuvor vom Funktionsbaustein UA_NodeGetHandle [▶ 122] ausgegeben wurde.

stNodeAddInfo : Definiert zusätzliche Informationen, z.B. welches Attribut aus dem UA Namensraum gelesen (Standard: 'Value' Attribut) oder welcher IndexRange verwendet werden soll. Wird durch struct ST_UANodeAdditionalInfo [▶ 131] spezifiziert.

pVariable : Zeiger auf Daten, wo die gelesen Daten gespeichert werden.

cbData : Bestimmt die Größe der zu lesenden Daten.

Timeout : Zeit bis zum Abbruch der Function. DEFAULT_ADS_TIMEOUT ist eine globale Konstante, gesetzt auf 5 Sekunden.

VAR_OUTPUT

```
VAR_OUTPUT
    Done        : BOOL;
    Busy        : BOOL;
    Error       : BOOL;
    ErrorID     : UDINT;
    cbData_R    : UDINT;
END_VAR
```

Done : Schaltet auf TRUE, wenn der Funktionsbaustein ausgeführt wurde.

Busy : Dieser Ausgang ist TRUE, bis der Baustein einen Befehl ausführt hat, maximal für die Dauer des 'Timeout' am Eingang. Solange Busy = TRUE akzeptieren die Eingänge keinen neuen Befehl. Es wird nicht die Zeit der Verbindung sondern die Empfangszeit überwacht.

Error : Dieser Ausgang schaltet auf TRUE, wenn bei der Ausführung eines Befehls ein Fehler auftritt. Der befehlsspezifische Fehlercode ist in ‚nErrID' enthalten.

ErrorID : Enthält den befehlsspezifischen ADS Fehlercode des zuletzt ausgeführten Befehls.

cbData_R : Anzahl zu lesender Bytes.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT 3.1	Win32, Win64, CE-X86, CE-ARM	Tc3_PLCopen_OpcUa
TwinCAT 2.11 R3 Build 2245	Win32, CE-X86, CE-ARM	Tc2_PLCopen_OpcUa

5.1.1.10 UA_Write

Dieser Funktionsbaustein schreibt Werte in ein gegebenes Knoten- und Verbindungshandle.

VAR_INPUT

```
VAR_INPUT
    Execute         : BOOL;
    ConnectionHdl   : DWORD;
    NodeHdl         : DWORD;
    stNodeAddInfo   : ST_UANodeAdditionalInfo;
```

```
    pVariable      : PVOID;
    cbData         : UDINT;
    Timeout        : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

Execute : Der Befehl wird durch eine steigende Flanke an diesem Eingang ausgelöst.

ConnectionHdl : Verbindungshandle, das vorher vom Funktionsbaustein UA_Connect [▶ 116] ausgegeben wurde.

NodeHdl : Knotenhandle, der zuvor vom Funktionsbaustein UA_NodeGetHandle [▶ 122] ausgegeben wurde.

stNodeAddInfo : Definiert zusätzliche Informationen, z.B. auf welchen IndexRange oder welches Attribut geschrieben werden soll (standardmäßig wird das 'Value' Attribut verwendet). Wird durch struct ST_UANodeAdditionalInfo [▶ 131] spezifiziert.

pVariable : Zeiger auf zu schreibende Daten.

cbData : Legt die Größe der zu schreibenden Werte fest.

Timeout : Zeit bis zum Abbruch der Function. DEFAULT_ADS_TIMEOUT ist eine globale Konstante, gesetzt auf 5 Sekunden.

VAR_OUTPUT

```
VAR_OUTPUT
    Done       : BOOL;
    Busy       : BOOL;
    Error      : BOOL;
    ErrorID    : DWORD;
END_VAR
```

Done : Schaltet auf TRUE, wenn der Funktionsbaustein ausgeführt wurde.

Busy : Dieser Ausgang ist TRUE, bis der Baustein einen Befehl ausführt hat, maximal für die Dauer des 'Timeout' am Eingang. Solange Busy = TRUE akzeptieren die Eingänge keinen neuen Befehl. Es wird nicht die Zeit der Verbindung sondern die Empfangszeit überwacht.

Error : Dieser Ausgang schaltet auf TRUE, wenn bei der Ausführung eines Befehls ein Fehler auftritt. Der befehlsspezifische Fehlercode ist in 'ErrorID' enthalten.

ErrorID : Enthält den befehlsspezifischen ADS Fehlercode des zuletzt ausgeführten Befehls.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT 3.1	Win32, Win64, CE-X86, CE-ARM	Tc3_PLCopen_OpcUa
TwinCAT 2.11 R3 Build 2245	Win32, CE-X86, CE-ARM	Tc2_PLCopen_OpcUa

5.1.2 Datentypen

5.1.2.1 E_UAAttributeID

E_UAAttributeID

```
TYPE E_UAAttributeID:
(
    eUAAI_NodeID     := 1,
    eUAAI_NodeClass    := 2,
    eUAAI_BrowseName := 3,
    eUAAI_DisplayName := 4,
    eUAAI_Description := 5,
    eUAAI_WriteMask    := 6,
    eUAAI_UserWriteMask := 7,
    eUAAI_IsAbstract := 8,
    eUAAI_Symmetric    := 9,
```

```
    eUAAI_InverseName := 10,
    eUAAI_ContainsNoLoops := 11,
    eUAAI_EventNotifier := 12,
    eUAAI_Value     := 13,
    eUAAI_DataType     := 14,
    eUAAI_ValueRank    := 15,
    eUAAI_ArrayDimensions := 16
)DINT;
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT 3.1	Win32, Win64, CE-X86, CE-ARM	Tc3_PLCopen_OpcUa
TwinCAT 2.11 R3 Build 2245	Win32, CE-X86, CE-ARM	Tc2_PLCopen_OpcUa

5.1.2.2 E_UADataType

E_UADataType

```
TYPE E_UADataType:
(
    eUAType_Undefinied := -1,
    eUAType_Null     := 0,
    eUAType_Boolean    := 1,
    eUAType_SByte     := 2,
    eUAType_Byte     := 3,
    eUAType_Int16     := 4,
    eUAType_UInt16     := 5,
    eUAType_Int32     := 6,
    eUAType_UInt32     := 7,
    eUAType_Int64     := 8,
    eUAType_UInt64     := 9,
    eUAType_Float     := 10,
    eUAType_Double     := 11,
    eUAType_String     := 12,
    eUAType_DateTime := 13,
    eUAType_Guid     := 14,
    eUAType_ByteString := 15,
    eUAType_XmlElement := 16,
    eUAType_NodeId     := 17,
    eUAType_ExpandedNodeId := 18,
    eUAType_StatusCode := 19,
    eUAType_QualifiedName := 20,
    eUAType_LocalizedText := 21,
    eUAType_ExtensionObject:= 22,
    eUAType_DataValue := 23,
    eUAType_Variant    := 24,
    eUAType_DiagnosticInfo := 25
)DINT;
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT 3.1	Win32, Win64, CE-X86, CE-ARM	Tc3_PLCopen_OpcUa
TwinCAT 2.11 R3 Build 2245	Win32, CE-X86, CE-ARM	Tc2_PLCopen_OpcUa

5.1.2.3 E_UAIdentifierType

E_UATransportProfile

```
TYPE E_UAIdentifierType:
(
    eUAIdentifierType_String := 1,
    eUAIdentifierType_Numeric := 2,
```

```
    eUAIdentifierType_GUID := 3,
    eUAIdentifierType_Opaque := 4
)DINT;
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT 3.1	Win32, Win64, CE-X86, CE-ARM	Tc3_PLCopen_OpcUa
TwinCAT 2.11 R3 Build 2245	Win32, CE-X86, CE-ARM	Tc2_PLCopen_OpcUa

5.1.2.4 E_UASecurityMsgMode

E_UASecurityMsgMode

```
TYPE E_UASecurityMsgMode:
(
    eUASecurityMsgMode_BestAvailable := 0,
    eUASecurityMsgMode_None      := 1,
    eUASecurityMsgMode_Sign      := 2,
    eUASecurityMsgMode_Sign_Encrypt := 3
)DINT;
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT 3.1	Win32, Win64, CE-X86, CE-ARM	Tc3_PLCopen_OpcUa
TwinCAT 2.11 R3 Build 2245	Win32, CE-X86, CE-ARM	Tc2_PLCopen_OpcUa

5.1.2.5 E_UASecurityPolicy

E_UASecurityPolicy

```
TYPE E_UASecurityPolicy:
(
    eUASecurityPolicy_BestAvailable := 0
    eUASecurityPolicy_None      := 1,
    eUASecurityPolicy_Basic128      := 2,
    eUASecurityPolicy_Basic128Rsa15 := 3,
    eUASecurityPolicy_Basic256      := 4
)DINT;
END_TYPE
```

None : Richtlinie für Konfigurationen mit geringsten Sicherheitsanforderungen.

Basic128 : Richtlinie für Konfigurationen mit geringen bis mittleren Sicherheitsanforderungen.

Basic128Rsa15 : definiert Sicherheitsrichtlinie für Konfigurationen mit mittlerer bis hoher Sicherheit.

Basic256 : definiert eine Sicherheitsrichtlinie für Konfigurationen mit hohen Sicherheitsanforderungen.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT 3.1	Win32, Win64, CE-X86, CE-ARM	Tc3_PLCopen_OpcUa
TwinCAT 2.11 R3 Build 2245	Win32, CE-X86, CE-ARM	Tc2_PLCopen_OpcUa

5.1.2.6 E_UATransportProfile

E_UATransportProfile

```
TYPE E_UATransportProfile:
(
    eUATransportProfileUri_UATcp        := 1,
    eUATransportProfileUri_WSHttpBinary     := 2,
    eUATransportProfileUri_WSHttpXmlOrBinary := 3,
    eUATransportProfileUri_WSHttpXml     := 4
)DINT;
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT 3.1	Win32, Win64, CE-X86, CE-ARM	Tc3_PLCopen_OpcUa
TwinCAT 2.11 R3 Build 2245	Win32, CE-X86, CE-ARM	Tc2_PLCopen_OpcUa

5.1.2.7 ST_UASessionConnectInfo

ST_UASessionConnectInfo

```
TYPE ST_UASessionConnectInfo:
STRUCT
    sApplicationUri     : STRING(MAX_STRING_LENGTH);
    sApplicationName    : STRING(MAX_STRING_LENGTH);

    eSecurityMode     : E_UASecurityMsgMode;
    eSecurityPolicyUri : E_UASecurityPolicy;
    eTransportProfileUri : E_UATransportProfile;

    tSessionTimeout     : TIME;
    tConnectTimeout     : TIME;
END_STRUCT
END_TYPE
```

sApplicationUri: Anwendungs-Uri, maximale Zeichenkettenlänge 255.
Ab TcUAClient 2.0.0.14 wird diese automatisch vom Zertifikat vorgegeben, wie in der PLCOpen Spezifikation definiert.

sApplicationName: Anwendungsname mit maximaler Zeichenkettenlänge von 255.

eSecurityMode: Sicherheitsmeldungsmodus. Siehe verfügbare Modi hier E_UASecurityMsgMode [▶ 128].

eSecurityPolicyUri: Sicherheitsrichtlinien-Uri. Siehe verfügbare Sicherheitsrichtlinien-uri's hier E_UASecurityPolicy [▶ 128].

eTransportProfileUri: Transportprofil-Uri. Siehe verfügbare Transportprofil-uri's hier E_UATransportProfile [▶ 129];

nSessionTimeout: Wert Sitzungstimeout.

nConnectTimeout: Wert Verbindungstimeout.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT 3.1	Win32, Win64, CE-X86, CE-ARM	Tc3_PLCopen_OpcUa
TwinCAT 2.11 R3 Build 2245	Win32, CE-X86, CE-ARM	Tc2_PLCopen_OpcUa

5.1.2.8 ST_UAIndexRange

ST_UAIndexRange

```
TYPE ST_UAIndexRange:
STRUCT
    nStartIndex     : UDINT;
    nEndIndex     : UDINT;
END_STRUCT
END_TYPE
```

Für alle Dimensionen:

- StartinIndex und EndIndex müssen zugewiesen werden.
- StartIndex muss kleiner als EndIndex sein
- Um auf alle Elemente in einer Dimension zugreifen zu können, müssen StartIndex und EndIndex abhängig von der Gesamtzahl Elemente in der Dimension zugewiesen werden.
- Einzelne Elemente einer Dimension können ausgewählt werden, indem der gleiche StartIndex und EndIndex angegeben wird.

nStartIndex : Startindex der Daten.

nEndIndex : Endindex der Daten.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT 3.1	Win32, Win64, CE-X86, CE-ARM	Tc3_PLCopen_OpcUa
TwinCAT 2.11 R3 Build 2245	Win32, CE-X86, CE-ARM	Tc2_PLCopen_OpcUa

5.1.2.9 ST_UAMethodArgInfo

ST_UAMethodArgInfo

```
TYPE ST_UAMethodArgInfo:
STRUCT
    DataType      : E_UADataType := -1;
    ValueRank     : DINT := 2147483647;
    ArrayDimensions : ARRAY[1..3] OF UDINT := [0,0,0];
    nLenData      : DINT;
END_STRUCT
END_TYPE
```

E_UADataType: Legt den UA Datentyp für den Methodenparameter fest. (E_UADataType [▶ 127])

ValueRank: Legt fest, ob der Parameter Skalar (-1) oder Array ist.

ArrayDimensions: Wenn der Parameter ein Array ist, spezifiziert dieser die Dimensionen des Array. Jedes Element bestimmt die Länge pro Dimension.

nLenData: Spezifiziert die Länge des Arguments. Bei Ausgabeinformationen wird von struct nur dieses Element gefordert.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT 3.1	Win32, Win64, CE-X86, CE-ARM	Tc3_PLCopen_OpcUa
TwinCAT 2.11 R3 Build 2245	Win32, CE-X86, CE-ARM	Tc2_PLCopen_OpcUa

5.1.2.10 ST_UANodeID

ST_UANodeID

```
TYPE ST_UANodeID:
STRUCT
    nNamespaceIndex     : UINT;
    nReserved     : ARRAY [1..2] OF BYTE; //fill bytes
    sIdentifier     : STRING(MAX_STRING_LENGTH);
    eIdentifierType     : E_UAIdentifierType;
END_STRUCT
END_TYPE
```

nNamespaceIndex : Namensraum-Index unter dem der Knoten verfügbar ist. Kann mit dem Funktionsbaustein UA_GetNamespaceIndex [▶ 118] bestimmt werden.

sIdentifier : Bezeichner wie im UA Namensraum gezeigt (Attribut 'Identifier').

eIdentifierType : Typ der Variablen, beschrieben mittels E_UAIdentifierType [▶ 127].

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT 3.1	Win32, Win64, CE-X86, CE-ARM	Tc3_PLCopen_OpcUa
TwinCAT 2.11 R3 Build 2245	Win32, CE-X86, CE-ARM	Tc2_PLCopen_OpcUa

5.1.2.11 ST_UANodeAdditionalInfo

ST_UANodeAdditionalInfo

```
TYPE ST_UANodeAdditionalInfo:
STRUCT
    eAttributeID : E_UAAttributeID;
    nIndexRangeCount : UINT;
    nReserved     : ARRAY[1..2] OF BYTE; // fill bytes
    stIndexRange : ARRAY[1..nMaxIndexRange] OF ST_UAIndexRange;
END_STRUCT
END_TYPE
```

eAttributeID: Spezifiziert die ID des OPC-UA Attributs. Standardmäßig wird eUAAI_Value verwendet. (E_UAAttributeID [▶ 126])

nIndexRangeCount: Legt fest, wie viele Indexbereiche in stIndexRange verwendet werden.

stIndexRange: Spezifiziert einen Indexbereich für das Lesen von Werten aus einem Array. (ST_UAIndexRange [▶ 130])

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT 3.1	Win32, Win64, CE-X86, CE-ARM	Tc3_PLCopen_OpcUa
TwinCAT 2.11 R3 Build 2245	Win32, CE-X86, CE-ARM	Tc2_PLCopen_OpcUa

6 Beispiele

6.1 Übersicht

Beispiel	Kategorie	TwinCAT-Version	Beschreibung	Download
1	OPC-UA Server	TC2 / TC3	Data Access: SPS-Beispielprogramm	Beispiel
2	OPC-UA Server	TC2 / TC3	Historical Access: SPS-Beispielprogramm	Beispiel
3	OPC-UA Client Tc3_PLCopen_OpcUa	TC3	Beispiele für UA_Read, UA_Write und UA_MethodCall	Beispiel
4	OPC-UA Client Tc2_PLCopen_OpcUa	TC2	Beispiele für UA_Read und UA_Write	Beispiel

7 Anhang

7.1 OPC-UA Server Fehlercodes

Die folgende Tabelle gibt eine Übersicht über die globalen OPC-UA Fehlermeldungen und ihre Bedeutung.

Hex	Description
0xE4DD0000	general UA Server error
0xE4DA0000	general ADS error
0xE4DB0000	general API error
0xE4DD0100	Server is already connected
0xE4DD0101	general error while establishing a connection to the UA server
0xE4DD0100	Server is already connected
0xE4DD0101	general error while establishing a connection to the UA server
0xE4DD0102	failed to setup UA security
0xE4DD0103	item ID already exists
0xE4DD0104	item does not exist
0xE4DD0105	invalid or unsupported item type
0xE4DD0106	var types cannot be converted
0xE4DD0107	device is suspended
0xE4DD0108	vartype conversion is not supported for this type
0xE4DD0109	a namespace with the requested name cannot be found
0xE4DD0110	target host not found
0xE4DD0111	timeout due to missing reply from target host
0xE4DD0112	UA Client session handle is invalid
0xE4DD0113	invalid node ID

7.2 OPC-UA Client Fehlercodes

Die zuvor beschriebenen Funktionsbausteine haben Ausgabevariable, die Fehler und eine ErrorID für weitere Informationen zum aufgetretenen Problem anzeigen.
Diese Fehlercodeliste führt die möglichen Werte auf.

Name	Beschreibung	Upper Bytes	Lower Bytes
UAC_E_CONNECTED	Server bereits verbunden	0xE4DD	0x0100
UAC_E_CONNECT	Allgemeiner Fehler beim Aufbau einer Verbindung		0x0101
UAC_E_UASECURITY	UA security konnte nicht eingerichtet werden		0x0102
UAC_E_ITEMEXISTS	Element ID bereits vorhanden		0x0103
UAC_E_ITEMNOTFOUND	Element existiert nicht		0x0104
UAC_E_ITEMTYPE	Ungültiger oder nicht unterstützter Elementtyp		0x0105
UAC_E_CONVERSION	Variablentypen können nicht konvertiert werden		0x0106
UAC_E_SUSPENDED	Gerät hängt. Bitte später erneut versuchen...		0x0107
UAC_E_TYPE_NOT_SUPPORTED	Konvertierung Variablentyp wird nicht unterstützt.		0x0108
UAC_E_NSNAME_NOTFOUND	Kein Namensraum mit dem angegebenen Namen gefunden.		0x0109
UAC_E_CONNECT_NOTFOUND	Verbindung fehlgeschlagen: Zielhost konnte nicht gefunden werden.		0x0110
UAC_E_TIMEOUT	Timeout: dh. Zielhost antwortet nicht		0x0111
UAC_E_INVALIDHDL	Sitzungshandle ungültig		0x0112
UAC_E_INVALIDNODEID	UA Knoten-ID unbekannt		0x0113
UAC_E_INVAL_IDENTIFIER_TYPE	Bezeichnertyp der UaNodeId ungültig		0x0114
UAC_E_IDENTIFIER_NOTSUPP	Bezeichnertyp UaNodeId wird nicht unterstützt		0x0115
UAC_E_INVAL_NODE_HDL	Ungültiges Knotenhandle		0x0116
UAC_E_UAREADFAILED	UA Read aus unbekannter Ursache fehlgeschlagen		0x0117
UAC_E_UAWRITEFAILED	UA Write aus unbekannter Ursache fehlgeschlagen		0x0118
UAC_E_INVAL_NODEMETHOD_HDL	Ungültiges Methodenhandle		0x0119
UAC_E_CALL_FAILED	Aufruf fehlgeschlagen, Ursache unbekannt		0x011A
UAC_E_CALLDECODE_FAILED	Aufruf erfolgreich, Decodierung Rückgabewert fehlgeschlagen		0x011B
UAC_E_NOTMAPPEDTYPE	Nicht zugeordneter Datentyp in Rückgabewert		0x011C
UAC_E_CALL_FAILED_BADINTERNAL	Aufruf fehlgeschlagen mit UA_BadInternal		0x011D

Name	Beschreibung	Upper Bytes	Lower Bytes
UAC_E_METHODIDINVALID	Unbekannte MethodenID (bei Aufruf zurückgegeben, auch wenn von GetMethodHdl bereitgestellt)		0x011E
UAC_E_TOOMUCHDIM	Methodenaufruf hat Parameter mit mehr als 3 Dimensionen zurückgegeben - wird nicht unterstützt.		0x011F
UAC_E_CALL_FAILED_INVALIDARG	Aufruf fehlgeschlagen mit OpcUa_BadInvalidArgument		0x0120
UAC_E_CALL_FAILED_TYPEMISMATCH	Aufruf fehlgeschlagen mit UAC_E_CALL_FAILED_TYPEMISMATCH		0x0121
UAC_E_CALL_FAILED_OUTOFRANGE	Aufruf fehlgeschlagen mit UAC_E_CALL_FAILED_OUTOFRANGE		0x0122
UAC_E_CALL_FAILED_BADSTRUCTURE	Aufruf fehlgeschlagen mit OpcUa_BadStructureMissing		0x0123
UAC_E_CALL_TYPEMISMATCH_OUTPARAM	Aufruf erfolgreich, aber keine Typenübereinstimmung der bereitgestellten Ausgabeinformation		0x0124
DEVICE_INVALIDSIZE	Parametergröße nicht korrekt		0x705
DEVICE_INVALIDDATA	Ungültige Parameterwerte		0x706

Siehe auch ADS Rückgabecodes [▶ 135] für weitere Rückgabecodes.

7.3 ADS Return Codes

Fehlercode: 0x000 [▶ 135]..., 0x500 [▶ 136]..., 0x700 [▶ 136]..., 0x1000 [▶ 138].....

Globale Fehlercodes

Hex	Dec	Beschreibung
0x0	0	Kein Fehler
0x1	1	Interner Fehler
0x2	2	Keine Echtzeit
0x3	3	Zuweisung gesperrt-Speicherfehler
0x4	4	Postfach voll
0x5	5	Falsches HMSG
0x6	6	Ziel-Port nicht gefunden
0x7	7	Zielrechner nicht gefunden
0x8	8	Unbekannte Befehl-ID
0x9	9	Ungültige Task-ID
0xA	10	Kein IO
0xB	11	Unbekannter ADS-Befehl
0xC	12	Win32 Fehler
0xD	13	Port nicht angeschlossen
0xE	14	Ungültige ADS-Länge
0xF	15	Ungültige AMS Net ID
0x10	16	niedrige Installationsebene
0x11	17	Kein Debugging verfügbar
0x12	18	Port deaktiviert
0x13	19	Port bereits verbunden
0x14	20	ADS Sync Win32 Fehler
0x15	21	ADS Sync Timeout
0x16	22	ADS Sync AMS Fehler
0x17	23	Keine Index-Map für ADS Sync vorhanden
0x18	24	Ungültiger ADS-Port
0x19	25	Kein Speicher
0x1A	26	TCP Sendefehler
0x1B	27	Host nicht erreichbar
0x1C	28	Ungültiges AMS Fragment

Router Fehlercodes

Hex	Dec	Name	Beschreibung
0x500	1280	ROUTERERR_NOLOCKEDMEMORY	Lockierter Speicher kann nicht zugewiesen werden.
0x501	1281	ROUTERERR_RESIZEMEMORY	Die Größe des Routerspeichers konnte nicht geändert werden.
0x502	1282	ROUTERERR_MAILBOXFULL	Das Postfach hat die maximale Anzahl der möglichen Meldungen erreicht. Die aktuell gesendete Nachricht wurde abgewiesen.
0x503	1283	ROUTERERR_DEBUGBOXFULL	Das Postfach hat die maximale Anzahl der möglichen Meldungen erreicht. Die gesendete Nachricht wird nicht im ADS Monitor angezeigt.
0x504	1284	ROUTERERR_UNKNOWNPORTTYPE	Der Porttyp ist unbekannt.
0x505	1285	ROUTERERR_NOTINITIALIZED	Router ist nicht initialisiert.
0x506	1286	ROUTERERR_PORTALREADYINUSE	Die gewünschte Portnummer ist bereits vergeben.
0x507	1287	ROUTERERR_NOTREGISTERED	Der Port ist nicht registriert.
0x508	1288	ROUTERERR_NOMOREQUEUES	Die maximale Anzahl von Ports ist erreicht.
0x509	1289	ROUTERERR_INVALIDPORT	Der Port ist ungültig.
0x50A	1290	ROUTERERR_NOTACTIVATED	Der Router ist nicht aktiv.

Allgemeine ADS Fehlercodes

Hex	Dec	Name	Beschreibung
0x700	1792	ADSERR_DEVICE_ERROR	Gerätefehler
0x701	1793	ADSERR_DEVICE_SRVNOTSUPP	Service wird vom Server nicht unterstützt
0x702	1794	ADSERR_DEVICE_INVALIDGRP	Ungültige Index-Gruppe
0x703	1795	ADSERR_DEVICE_INVALIDOFFSET	Ungültiger Index-Offset
0x704	1796	ADSERR_DEVICE_INVALIDACCESS	Lesen und schreiben nicht gestattet.
0x705	1797	ADSERR_DEVICE_INVALIDSIZE	Parametergröße nicht korrekt
0x706	1798	ADSERR_DEVICE_INVALIDDATA	Ungültige Parameter-Werte
0x707	1799	ADSERR_DEVICE_NOTREADY	Gerät ist nicht betriebsbereit
0x708	1800	ADSERR_DEVICE_BUSY	Gerät ist beschäftigt
0x709	1801	ADSERR_DEVICE_INVALIDCONTEXT	Ungültiger Kontext (muss in Windows sein)
0x70A	1802	ADSERR_DEVICE_NOMEMORY	Nicht genügend Speicher
0x70B	1803	ADSERR_DEVICE_INVALIDPARM	Ungültige Parameter-Werte
0x70C	1804	ADSERR_DEVICE_NOTFOUND	Nicht gefunden (Dateien,...)
0x70D	1805	ADSERR_DEVICE_SYNTAX	Syntax-Fehler in Datei oder Befehl
0x70E	1806	ADSERR_DEVICE_INCOMPATIBLE	Objekte stimmen nicht überein
0x70F	1807	ADSERR_DEVICE_EXISTS	Objekt ist bereits vorhanden
0x710	1808	ADSERR_DEVICE_SYMBOLNOTFOUND	Symbol nicht gefunden
0x711	1809	ADSERR_DEVICE_SYMBOLVERSIONINVALID	Symbol-Version ungültig
0x712	1810	ADSERR_DEVICE_INVALIDSTATE	Gerät im ungültigen Zustand
0x713	1811	ADSERR_DEVICE_TRANSMODENOTSUPP	AdsTransMode nicht unterstützt
0x714	1812	ADSERR_DEVICE_NOTIFYHNDINVALID	Notification Handle ist ungültig
0x715	1813	ADSERR_DEVICE_CLIENTUNKNOWN	Notification-Client nicht registriert
0x716	1814	ADSERR_DEVICE_NOMOREHDLS	Keine weitere Notification Handle
0x717	1815	ADSERR_DEVICE_INVALIDWATCHSIZE	Größe der Notification zu groß
0x718	1816	ADSERR_DEVICE_NOTINIT	Gerät nicht initialisiert
0x719	1817	ADSERR_DEVICE_TIMEOUT	Gerät hat einen Timeout
0x71A	1818	ADSERR_DEVICE_NOINTERFACE	Interface Abfrage fehlgeschlagen
0x71B	1819	ADSERR_DEVICE_INVALIDINTERFACE	Falsches Interface angefordert
0x71C	1820	ADSERR_DEVICE_INVALIDCLSID	Class-ID ist ungültig
0x71D	1821	ADSERR_DEVICE_INVALIDOBJID	Object-ID ist ungültig
0x71E	1822	ADSERR_DEVICE_PENDING	Anforderung steht aus
0x71F	1823	ADSERR_DEVICE_ABORTED	Anforderung wird abgebrochen
0x720	1824	ADSERR_DEVICE_WARNING	Signal-Warnung
0x721	1825	ADSERR_DEVICE_INVALIDARRAYIDX	Ungültiger Array-Index
0x722	1826	ADSERR_DEVICE_SYMBOLNOTACTIVE	Symbol nicht aktiv
0x723	1827	ADSERR_DEVICE_ACCESSDENIED	Zugriff verweigert
0x724	1828	ADSERR_DEVICE_LICENSENOTFOUND	Fehlende Lizenz
0x725	1829	ADSERR_DEVICE_LICENSEEXPIRED	Lizenz abgelaufen
0x726	1830	ADSERR_DEVICE_LICENSEEXCEEDED	Lizenz überschritten
0x727	1831	ADSERR_DEVICE_LICENSEINVALID	Lizenz ungültig
0x728	1832	ADSERR_DEVICE_LICENSESYSTEMID	Lizenz der System-ID ungültig
0x729	1833	ADSERR_DEVICE_LICENSENOTIMELIMIT	Lizenz nicht zeitlich begrenzt
0x72A	1834	ADSERR_DEVICE_LICENSEFUTUREISSUE	Lizenzproblem: Zeitpunkt in der Zukunft
0x72B	1835	ADSERR_DEVICE_LICENSETIMETOLONG	Lizenz-Zeitraum zu lang
0x72c	1836	ADSERR_DEVICE_EXCEPTION	Exception beim Systemstart
0x72D	1837	ADSERR_DEVICE_LICENSEDUPLICATED	Lizenz-Datei zweimal gelesen
0x72E	1838	ADSERR_DEVICE_SIGNATUREINVALID	Ungültige Signatur
0x72F	1839	ADSERR_DEVICE_CERTIFICATEINVALID	öffentliches Zertifikat
0x740	1856	ADSERR_CLIENT_ERROR	Clientfehler
0x741	1857	ADSERR_CLIENT_INVALIDPARM	Dienst enthält einen ungültigen Parameter
0x742	1858	ADSERR_CLIENT_LISTEMPTY	Polling-Liste ist leer
0x743	1859	ADSERR_CLIENT_VARUSED	Var-Verbindung bereits im Einsatz
0x744	1860	ADSERR_CLIENT_DUPLINVOKEID	Die aufgerufene ID ist bereits in Benutzung
0x745	1861	ADSERR_CLIENT_SYNCTIMEOUT	Timeout ist aufgetreten
0x746	1862	ADSERR_CLIENT_W32ERROR	Fehler im Win32 Subsystem
0x747	1863	ADSERR_CLIENT_TIMEOUTINVALID	Ungültiger Client Timeout-Wert
0x748	1864	ADSERR_CLIENT_PORTNOTOPEN	ADS-Port nicht geöffnet
0x750	1872	ADSERR_CLIENT_NOAMSADDR	Interner Fehler in Ads-Sync

Hex	Dec	Name	Beschreibung
0x751	1873	ADSERR_CLIENT_SYNCINTERNAL	Hash-Tabelle-Überlauf
0x752	1874	ADSERR_CLIENT_ADDHASH	Schlüssel nicht gefunden im Hash
0x753	1875	ADSERR_CLIENT_REMOVEHASH	Keine weitere Symbole im Cache
0x754	1876	ADSERR_CLIENT_NOMORESYM	Ungültige Antwort empfangen
0x755	1877	ADSERR_CLIENT_SYNCRESINVALID	Sync Port ist gesperrt

RTime Fehlercodes

Hex	Dec	Name	Beschreibung
0x1000	4096	RTERR_INTERNAL	Interner Fehler im TwinCAT Echtzeit-System.
0x1001	4097	RTERR_BADTIMERPERIODS	Timer-Wert ist nicht gültig.
0x1002	4098	RTERR_INVALIDTASKPTR	Task-Pointer hat den ungültigen Wert 0 (null).
0x1003	4099	RTERR_INVALIDSTACKPTR	Task Stackpointer hat den ungültigen Wert 0.
0x1004	4100	RTERR_PRIOEXISTS	Die Request Task Priority ist bereits vergeben.
0x1005	4101	RTERR_NOMORETCB	Kein freies TCB (Task Control Block) zur Verfügung. Maximale Anzahl von TCBs beträgt 64.
0x1006	4102	RTERR_NOMORESEMAS	Keine freien Semaphoren zur Verfügung. Maximale Anzahl der Semaphoren beträgt 64.
0x1007	4103	RTERR_NOMOREQUEUES	Kein freier Platz in der Warteschlange zur Verfügung. Maximale Anzahl der Plätze in der Warteschlange beträgt 64.
0x100D	4109	RTERR_EXTIRQALREADYDEF	Ein externer Synchronisations-Interrupt wird bereits angewandt.
0x100E	4110	RTERR_EXTIRQNOTDEF	Kein externer Synchronisations-Interrupt angewandt.
0x100F	4111	RTERR_EXTIRQINSTALLFAILED	Anwendung des externen Synchronisierungs- Interrupts ist fehlgeschlagen
0x1010	4112	RTERR_IRQLNOTLESSOREQUAL	Aufruf einer Service-Funktion im falschen Kontext
0x1017	4119	RTERR_VMXNOTSUPPORTED	Intel VT-x Erweiterung wird nicht unterstützt.
0x1018	4120	RTERR_VMXDISABLED	Intel VT-x Erweiterung ist nicht aktiviert im BIOS.
0x1019	4121	RTERR_VMXCONTROLSMISSING	Fehlende Funktion in Intel VT-x Erweiterung.
0x101A	4122	RTERR_VMXENABLEFAILS	Aktivieren von Intel VT-x schlägt fehl.

TCP Winsock-Fehlercodes

Hex	Dec	Name	Beschreibung
0x274c	10060	WSAETIMEDOUT	Verbindungs Timeout aufgetreten. Fehler beim Herstellen der Verbindung, da die Gegenstelle nach einer bestimmten Zeitspanne nicht ordnungsgemäß reagiert hat, oder die hergestellte Verbindung konnte nicht aufrecht erhalten werden, da der verbundene Host nicht reagiert hat.
0x274d	10061	WSAECONNREFUSED	Verbindung abgelehnt. Es konnte keine Verbindung hergestellt werden, da der Zielcomputer dies explizit abgelehnt hat. Dieser Fehler resultiert normalerweise aus dem Versuch, eine Verbindung mit einem Dienst herzustellen, der auf dem fremden Host inaktiv ist—das heißt, einem Dienst, für den keine Serveranwendung ausgeführt wird.
0x2751	10065	WSAEHOSTUNREACH	Keine Route zum Host Ein Socketvorgang bezog sich auf einen nicht verfügbaren Host.
			Weitere Winsock-Fehlercodes: Win32-Fehlercodes

8 Third Party Beispiele

8.1 Beispiel OPC UA Client von Inosoft (Third Party)

Download Inosoft Beispiel "Boiler"

Download PLC Demo "Boiler"

Kochbuch:

1. Laden Sie das PLC-Demo "Boiler" in die TwinCAT SPS
2. Installieren Sie den TwinCAT-OPC-UA Server (XP oder CE je nach eingesetzter Plattform)
3. Entpacken Sie das Third-Party Demo Inosoft
4. Öffnen Sie die Datei "INOSOFT_UABoilerDemo_PC_800_600.vwn" mit Notepad
5. Suchen Sie die Zeile <Channel Name="Ch1" Type="OPCUA" ProgID="Beckhoff OPCUA" ComputerName="opc.tcp://188.100.13.207:4840">
6. Passen Sie die IP-Adresse an - hier muss die IP Adresse des Gerätes mit dem UA-Servers eingetragen werden
7. Speichern Sie die Datei und starten Sie die Applikation "INOSOFT_UABoilerDemo_PC_800_600.exe"
8. Bestätigen Sie den Demo-Mode mit OK

⇨ das Demo ist jetzt 1h lauffähig